בריאן המבלינג

ניהול איכות תוכנה

ניתוח מערכות עכשווי
ואיך עושים זאת על פי:

• ISO 9000-3
• ת"י 3–2000







הוצאת הוד-עמי
לספרי מחשבים

ניהול
איכות תוכנה

ISO 9000-3



כולל

ת"י 2000-3

תרגום לעברית: **אליעזר ארז**

עורך: **יצחק עמיהוד**

ייעוץ מקצועי: **דר' אביגדור זוננשטיין - מכון התקנים הישראלי**

עריכה לשונית ועיצוב: **אניקה סואץ, קרן קוברובסקי**

עיצוב עטיפה: **סטודיו מצגר**

שמות מסחריים

שמות המוצרים והשירותים המוזכרים בספר הינם שמות מסחריים רשומים של החברות שלהם. הוצאת McGraw-Hill והוצאת הוד-עמי עשו כמיטב יכולתן למסור מידע אודות השמות המסחריים המוזכרים בספר זה ולציין את שמות החברות, המוצרים והשירותים. שמות מסחריים רשומים (registered trademarks) המוזכרים בספר צוינו בהתאמה.

הודעה

ספר זה מיועד לתת מידע אודות מוצרים שונים. נעשו מאמצים רבים לגרום לכך שהספר יהיה שלם ואמין ככל שניתן, אך אין משתמעת מכך כל אחריות שהיא.

המידע ניתן "כמות שהוא" ("as is"). הוצאת McGraw-Hill והוצאת הוד-עמי אינן אחראיות כלפי יחיד או ארגון עבור כל אובדן או נזק אשר ייגרם, אם ייגרם, מהמידע שבספר זה, או מהדיסקט שעשוי להיות מצורף לו.

לשם שטף הקריאה כתוב ספר זה בלשון זכר בלבד. ספר זה מיועד לגברים ונשים כאחד ואין בכוונתנו להפלות או לפגוע בציבור המשתמשים/ות.

שירותי תקשורת:

- **טלפון: 09-9564716 24 שעות**
- **פקס: 09-9571582 24 שעות**
- **חנות וירטואלית באינטרנט: *http://www.exlmarket.co.il/***
- **דואר אלקטרוני: *hod_ami@netvision.net.il***

ניהול
איכות תוכנה

ISO 9000-3



כולל

ת"י 2000-3

בריאן המבלין



הוצאת הוד-עמי
לספרי מחשבים



Managing Software Quality

By Brian Hambling

Hebrew Translation: **E. Erez**

Editor: **I. Amihud**

Authorized translation from the English language edition
published by McGraw-Hill, Copyright © 1996.
Hod-Ami Ltd. Copyright © 1997

(C)

כל הזכויות שמורות

הוצאת הוד-עמי
לספרי מחשבים בע"מ
ת.ד. 6108 הרצליה 46160
טלפון : 09-9564716 פקס : 09-9571582

אין להעתיק או לשדר בכל אמצעי שהוא ספר זה או קטעים ממנו בשום צורה ובשום אמצעי אלקטרוני או מכני, לרבות צילום והקלטה, אמצעי אחסון והפצת מידע, ללא אישור בכתב מאת ההוצאה, אלא לשם ציטוט קטעים קצרים בציון שם המקור.

הודפס בישראל
ניסן תשנ"ז, אפריל 1997

All Rights Reserved
HOD-AMI Ltd.
P.O.B. 6108, Herzliya
ISRAEL, April 1997

ISBN 965-361-124-0 מסת"ב

תוכן עניינים קצר

תוכן העניינים

פרק 2: מערכות לניהול האיכות - לשם מה 41

חלק 2 57

פרק 3: ניהול איכות, סיכונים ופרויקטים 59

פתח דבר

התוכנה כבשה לה מקום מרכזי בחיינו. השימוש בה כמרכיב החכם במוצרים, כאמצעי לשכלולם ולפישוטם של תהליכי הספקת השירות וכמקור להספקת מידע עדכני אמין ומנותח עבור הצרכים הניהוליים, הולך וגובר בקצב מסחרר.

במקביל, ולהתפתחות התוכנה חלק בזה, מתרחש תהליך הגלובליזציה שבו העולם הופך לכפר גדול אחד. תהליך זה מתבטא במיזוג של חברות והקמת חברות רב-לאומיות, במעבר כמעט חופשי של סחורות מכל אתר לכל אתר על פני הגלובוס, ובהספקת מידע עדכני בזמן אמת על כל נושא שהוא, לרבות חדשנותם ואיכותם של מוצרים חדשים וותיקים.

מצב זה מהווה כר פורה להגברת התחרות, כש"האיכות" מהווה פרמטר מרכזי בה, ולהגדלת מודעותו של הצרכן, הן כארגון והן כאדם בודד, לאיכותם של המוצרים. ואיכות נתפסת במגוון התכונות והאפשרויות שגלומות במוצר, ובמספר מינימלי של תקלות, לאורך מחזור חייו.

מצב עניינים זה מציב למגזר בתי התוכנה רף יעדים שמורם על ידי הלקוחות באופן תדיר, מדי יום ביומו, והרף כולל שתי דרישות, שעל פניו, בתפיסת הדברים הקלאסית, נראות כסותרות: הספקת תוכנות מורכבות יותר ובזמן תגובה קצר יותר מחד גיסא, ומסירתן כשהן "נקיות" מתקלות מאידך גיסא.

הניסיון, שנצבר תחילה בארגונים התעשייתים, לימד, שתנאי הכרחי לשיפור האיכות הינו הקמה וקיום של מערכת לניהול האיכות בארגון. בעולם שבו כל ספק הוא גם לקוח, התאגדו כולם ותחת החסות של ארגון התקינה הבינלאומי, פרסמו את סדרת התקנים בקבוצת ISO 9000. תקנים אלה זכו להצלחה מסחררת והפכו לרבי מכר.

כיום, הדרישה להתאמה לתקן הרלוונטי בסדרה זו, הוא תנאי הכרחי לקבלתה של הזמנת עבודה במרבית ההתקשרויות החוזיות.

בגלל ייחודה של התוכנה כמוצר, הוקדש לה תקן מיוחד שמספרו ISO 9000-3. תקן זה קיבע את הדרישות לניהול האיכות בארגונים שעיסוקם פיתוח תוכנות.

ההצלחה הכללית של ISO 9000, לא פסחה על תקן ייחודי זה. עדות לכך הוא מספרם הגדל במהירות של בתי התוכנה, בארץ ובעולם, שמקבלים את אישור ההתאמה לתקן.

אני מקווה שהקריאה בספר זה יהיה בה כדי לפתוח בפניך זוויות מבט חדשות בכל הקשור לניהול האיכות בתהליכי פיתוח התוכנה, ושהמידע והרעיונות שמצויים בו יסייעו לך ולארגונך בשיפור מערכת האיכות, בפיתוחם והפקתם של מוצרי תוכנה איכותיים יותר, וכתוצאה מכך, בשיפור כושר התחרותיות של הארגון.

זיוה פתיר

מנכ"ל מכון התקנים הישראלי

מבוא

הספר, ומה שיש בו

ספר זה מציין מסע של תגליות. לאחר קריירה בחומרה, בתוכנה ובהנדסת מערכות, בהם מילאה **האיכות** תפקיד משני ולא בולט, גויס המחבר לשורות חברת TickIT Auditors. בתפקיד בלתי מוכר ומלא אתגרים זה, התגלו לו דברים חדשים בקצב דחוס ומהיר. מעטות ההתנסויות שהן מאירות עיניים יותר מאשר בדיקה מפורטת של שגיאות הזולת, בהן אתה נוכח שהן בדיוק אותן שגיאות שעשית אתה במצבים דומים רק לפני שנים מספר. התחושה, שבמקצוע המאופיין בקצב גבוה של שינויים טכנולוגיים, דווקא הדברים החשובים באמת הם המנוגדים לשינויים ולשיפורים כאחד, היא ממש מהממת.

המסע האמיתי מתבטא בהכרה ההדרגתית **שהנדסת תוכנה** מעשית **וניהול האיכות** אינם בלתי תואמים. היפוכו של דבר, רבות מהבעיות המעיקות עלינו, שורשיהן נעוצים בפיתוח גרוע של **תרבות האיכות**, בעוד אנו מחפשים את הפתרונות בטכנולוגיה. דוגמה קלאסית לכך ישמשו שפע הכלים לניהול פרויקטים, שנועדו להפוך את התהליכים לאוטומטיים, ואשר רק מהנדסי תוכנה מעטים הצליחו ללמוד את השימוש בהם.

קריירה בענף התוכנה, **שהתחלתה** בניהול צוות פיתוח תוכנה אינה דבר שכיח, אך יש גם יתרונות ברכישת ניסיון מעשי לפני רכישת השכלה. כאשר באה לידי ההזדמנות להעביר שנה בלימודים שלאחר התואר, היתה זאת חוויה רצויה ופורייה. שהות מסוימת בתעשיה יכולה להוות תרופת-נגד לשגיונות של חיי האוניברסיטה ועימות קשה לרעיונות החדשים שטופחו שם. ההתנסות שלי בניהול פרויקטים בתעשיה הותירה בי עקבות שלא יימחו, ועוררה מספר שאלות יסוד בנוגע לקשר שבין האיכות לבין פרמטרים מוחשיים יותר של פרויקטים, כגון פרמטר העלויות. החזרה לחיים האוניברסיטאיים סיפקה את פסק הזמן שנדרש להרהור בהתנסויות שלי ולעיצוב מספר רעיונות לצורך הצגתם לתלמידים. בחמש השנים האחרונות עובדו רעיונות אלה ונוסו בשאון וההמולה של עבודת הייעוץ. בדומה למחברים רבים לפני, אני מבין עכשיו שאני יודע הרבה פחות ממה שחשבתי, ויש לי גם מספר רב של שאלות שנותרו ללא מענה.

מה שהתכוונתי לספק כאן, הוא **מורה דרך שימושי לניהול האיכות בענף התוכנה**. אראה סיפוק במאמצי, אם יעלה בידי להרתיע מביורוקרטיה מיותרת, או לשכנע מספר מהנדסי תוכנה להאמין שאכן **יש חשיבות לאיכות**. ספר זה אמור להיות נגיש למנהלים ולאנשי מקצוע בתוכנה ובאיכות, משום שכל אחד מהם ימצא בו חומר המכוון אליו ספציפית. אחת המטרות החשובות ביותר היא טיפוח הדו-שיח בין כל העוסקים **בניהול איכות התוכנה** - Software quality management. תקוותי היא, שכל שלוש הקבוצות יקראו בו וידונו בהשלכותיו.

הספר צמח מתוך TickIT Scheme, שהיא יוזמת איכות בריטית, אך הנושאים הנדונים הם אלה המתעוררים כתוצאה מיישום התקנים **ISO 9001** ו- **ISO 9000-3**, או מיישום תקנים והנחיות אחרים לניהול מערכת איכות. הספר אמור להיות בעל ערך בכל מקום בו קיימת גישה לאיכות המבוססת על תקנים, או במקום בו שוקלים שימוש בגישה זו.

בספר חמישה חלקים. **חלק ראשון** (המכיל שני פרקים), מספק חומר רקע והדרכה לרעיונות הבסיסיים שמאחורי הנדסת תוכנה, איכות תוכנה וניהול האיכות, ומבהיר במה עוסק ניהול מערכת איכות. פרק 1 מציג את הנושאים הראשיים בזה אחר זה ומציב אותם בהקשר שביניהם. כן מוצגות הגישות החלופיות להנדסת איכות התוכנה שנדון בהן בהמשך הספר. פרק 2 שוקל את המשמעות של תוכניות איכות והישגיהן. מוצגים עקרונות ניהול האיכות ומוסברת בו הגישה של ניהול מערכות איכות. יש אף התייחסות קצרה לתקן **ISO 9001** ולקווים המנחים של תקן **ISO 9000-3**.

תקנים אלה הותאמו לדרישות בישראל על ידי מכון התקנים הישראלי ויצאו לאור רשמית, ***ת"י* ISO 9001** ו-***ת"י 3-2000***, בהתאמה.

מן הראוי לציין שתקן **ISO 9000-3** שבו אנו דנים, הינו במהדורת 1991. ועדות התקינה של ISO מכינות מהדורה מעודכנת, שצפוייה לשנת 1998. התקן הישראלי המקביל יעודכן גם הוא וייקרא: ***ת"י* 3-ISO 9000**.

חלק שני מתמקד בשלושה רכיבים ראשיים של מערכת ניהול איכות: ניהול פרויקטים, פיתוח תוכנה וניהול האיכות. פרק אחד לכל אחד מנושאים אלה מזהה את הדיסציפלינות החיוניות ודן בקווים המנחים של תקן ISO 9000-3. לפני מתן עצות שימושיות בדבר הדרכים לטיפול בנושאים העיקריים בסביבה מסורתית, בה תהליך הפיתוח הוא ביסודו רציף, או קווי (ליניארי). פרק 3 בוחן את אופי הסיכונים שבפיתוח תוכנה ומזהה אחדים מהסיכונים הראשיים המשפיעים על פרויקטי פיתוח. הרעיונות הראשיים של ניהול פרויקטים ותכנון האיכות מתוארים בהקשר של ניהול הסיכונים. פרק 4 מתבונן בתהליך הפיתוח הבסיסי על ידי בחינת מחזור החיים והפעילויות המרכיבות אותו, בעוד פרק 5 בודק את נושאי ניהול האיכות שבתקן ISO 9000-3 ומציב אותם בהקשר שלהם עם ארגון לפיתוח תוכנה. פרק זה דן גם בצדדים המעשיים ליישום **לב תוכנית האיכות** (core quality system).

חלק שלישי עוסק בשאלת המדידה ושיפור תהליכים, ובו רק פרק אחד, פרק 6. הוא משלים את הצגת הקווים המנחים של תקן ISO 9000-3 על ידי המחשת מה שאמור להיות רמה מזערית קבילה של המדידה. הוא חוקר את יתרונות השימוש במדידות לשיפור מערכת ניהול האיכות, ומצביע קדימה אל היבט חדש של האיכות, אל מעבר

לתקן ISO 9000-3. הצעד הבא הוא טיפול במנגנון לשיפור תהליכים, שמאפשר לנו להמשיך ולשפר את ביצועי המערכת.

חלק רביעי דן בכמה מהכיוונים החדשים יותר של טכנולוגיית התוכנה, בהם תהליך הפיתוח הוא לא-קווי (לא-ליניארי), ושוקל אם ובאיזה מידה תקן ISO 9000-3 חל על טכנולוגיות אלו. הטכנולוגיות החדשות מציבות תביעות שונות למערכת ניהול איכות, וטכניקות האיכות שפותחו לפי ההנחיות של תקן ISO 9000-3 עשויות לא להתאים יותר. ניתן כאן ייעוץ מעשי בנושא ניהול האיכות לגבי טכנולוגיות לא-קוויות אלו. חלק זה מכיל שלושה פרקים: פרק 7 מציג את רעיון אי-הקוויות ומזהה את סוגי השיטות המאמצות את גישת הפיתוח הזו. מוצגת כאן שיטת מדידה של שיטות לא קוויות יחד עם מספר רעיונות בדבר אתגרי האיכות המחייבים טיפול. פרק 8 בוחן את גישות השימוש **באבטיפוסים** ובעיקר, את קבוצת הגישות שנקראת **פיתוח יישומים מהיר** (Rapid Application Development). הוא בוחן את הרציונל שמאחורי שיטות אלו ואת מאפייניהן המעשיים, לפני המעבר לפיתוח מספר עקרונות איכות שעליהם מבוססת אסטרטגיית ניהול האיכות המעשית עבור שיטות אלו. פרק 9 דן בפיתוח מוכוון-עצמים, גם כטכנולוגיה חשובה בזכות עצמה וגם כדוגמה לשיטת פיתוח שאינה רק איטרטיבית או רק רציפה, אלא, כפי שמכנים אותה בצדק, שיטת כלאיים. הוא בוחן את סוגיות האיכות המעשיות שבפיתוח כלאיים בכלל, ובפיתוח מוכוון-עצמים בפרט.

חלק חמישי, פרק 10, מאחד את הנושאים שנדונו בחלקים הקודמים של הספר. כך ניתן להגיע להערכה כוללת של תקן ISO 9000-3 ולאומדן תרומתו האפשרית בעתיד, לפני המעבר הלאה, לחיפוש גישת איכות שתתמוך בכל הטכנולוגיות. מהו סוג הגישה לניהול האיכות שיוכל להתאים לכל השיטות שהצגנו? האם יש קו משותף לכל השיטות מהעבר ומההווה שיוסיף להמשיך ולהתקיים? האם יש עקרונות ניהול איכות נצחיים שיש לנסות להעניק להם מעמד של קדושה?

לקוראי ספר זה אמורה להיות היכרות מעניינת עם פיתוח תוכנה מתוך היבט של ניהול איכות. לא נדרשת התמחות מיוחדת, משום שהחלק הראשון בספר פותח בהדרכה מספקת בתחום זה.

קוראים שאינם מכירים כלל מערכות ניהול איכות, או את נושא פיתוח התוכנה, רצוי שיתחילו בחלק הראשון וימשיכו ברצף לכל אורך הספר. אלה הבקיאים כבר במערכות ניהול איכות ובתקן ISO 9000-3 יכולים לדלג ללא כל חשש על חלק זה בזמן הקריאה הראשונה ולרפרף על החלק השני. עם זאת, עליהם להיות מודעים לעובדה שהגישה לאיכות מוגדרת בחלק הראשון.

אלה המעוניינים בעיקר ב**שיטות לפיתוח תוכנה** צריכים להתרכז בפרקים 1, 4, ו- 6 עד 9; **לניהול האיכות** יש להתרכז בפרקים 1, 2, 5, ו- 6 עד 9; **לניהול פרויקטים** יש להתרכז בפרקים 1, 4, ו- 6 עד 9. עם זאת, דברים אלה נכונים רק לגבי קריאה בלבד. כוחו של הספר בהיותו יחידה אחת שלמה.

בכל חלק של הספר, מוצגים תחילה הרעיונות בקטע המבוא. הרעיונות נתמכים ומודגמים במפת תפיסה כללית של הרעיונות ושל הקשרים שביניהם.

הערה: בשל הקושי לתרגם מונחים מסוימים, תמצא לעיתים מונחים מתורגמים אחרת מאשר אתה מכיר. לדוגמה: המילה practice תורגמה לנוהג, נהגי-, נהגים.

הפן הישראלי של תקני האיכות ופעילויות לאיכות

לספר שני נספחים:

נספח א׳: ת״י 2000-3

התקן הישראלי ת״י 2000-3 מקביל לתקן ISO 9000-3 שמוצג בספר. פרט למספר התאמות שנדרשו עבור תנאי הארץ, אין כל הבדל בין התקנים, וגם **מספרי הסעיפים נשארו ללא שינוי.** על כן, הקורא יכול למצוא את דרכו בתקן הישראלי בקלות רבה, בעת הקריאה בספר זה.

נספח ב׳: הפרס הלאומי לאיכות בתעשיה

בנספח זה תמצא את עיקרי התקנון למתן הפרס, אשר כולל את מטרות הענקתו, המדדים ובעיקר את פרטי טופס ההערכה שבו מפורטים הקריטריונים לבחינת זכאות לפרס.

מפת התפיסה הכללית

מפות המציגות את **התפיסה הכללית** (mind maps) הן טכניקה גרפית רבת עוצמה לרישום מידע בדרך טבעית לחשיבה האנושית. למפות אלו שימושים רבים, אך בספר זה הן משמשות **לתיאור תמציתי של הקשרים שבין רעיונות הקשורים זה לזה** ומוצגים בחלקים שונים של הספר.

למפת התפיסה הכללית ארבעה מאפיינים עיקריים:

1. **הנושא** בו מתרכזת תשומת הלב מגובש בדמות מרכזית אחת.
2. **הרעיונות הראשיים** של הנושא מסתעפים מהתמונה המרכזית בצורת ענפים.
3. הענפים מכילים **תמונת מפתח, או מילת מפתח, המודפסות על גבי קו הקישור.** נושאים בעלי חשיבות פחותה יותר מיוצגים כענפים הנספחים אל ענפים של רמה גבוהה יותר.
4. הענפים מהווים מבנה של **צמתים מחוברים.**

בתרשים 1 מוצגת מפת תפיסה כללית של הספר, כדי להראות כיצד הרעיונות המוצגים בספר קשורים זה לזה. מפות תפיסה (mind maps) דומות מופיעות כמורי דרך בתחילת כל חלק ספר.

בספר זה השתמשנו במפות תפיסה כדי להציג קישור של רעיונות הקשורים זה לזה. מפות התפיסה שנמצאות בתחילת כל חלק של הספר מכוונות לשמש רק כנקודת מוצא, שתעודד את הקוראים לחשוב על הקישורים המוצגים ולחפש קישורים חדשים. המפה לא נועדה לשמש כפעלול או כתרגיל אקדמי; אמיתות חשובות רבות מתגלות לעין רק

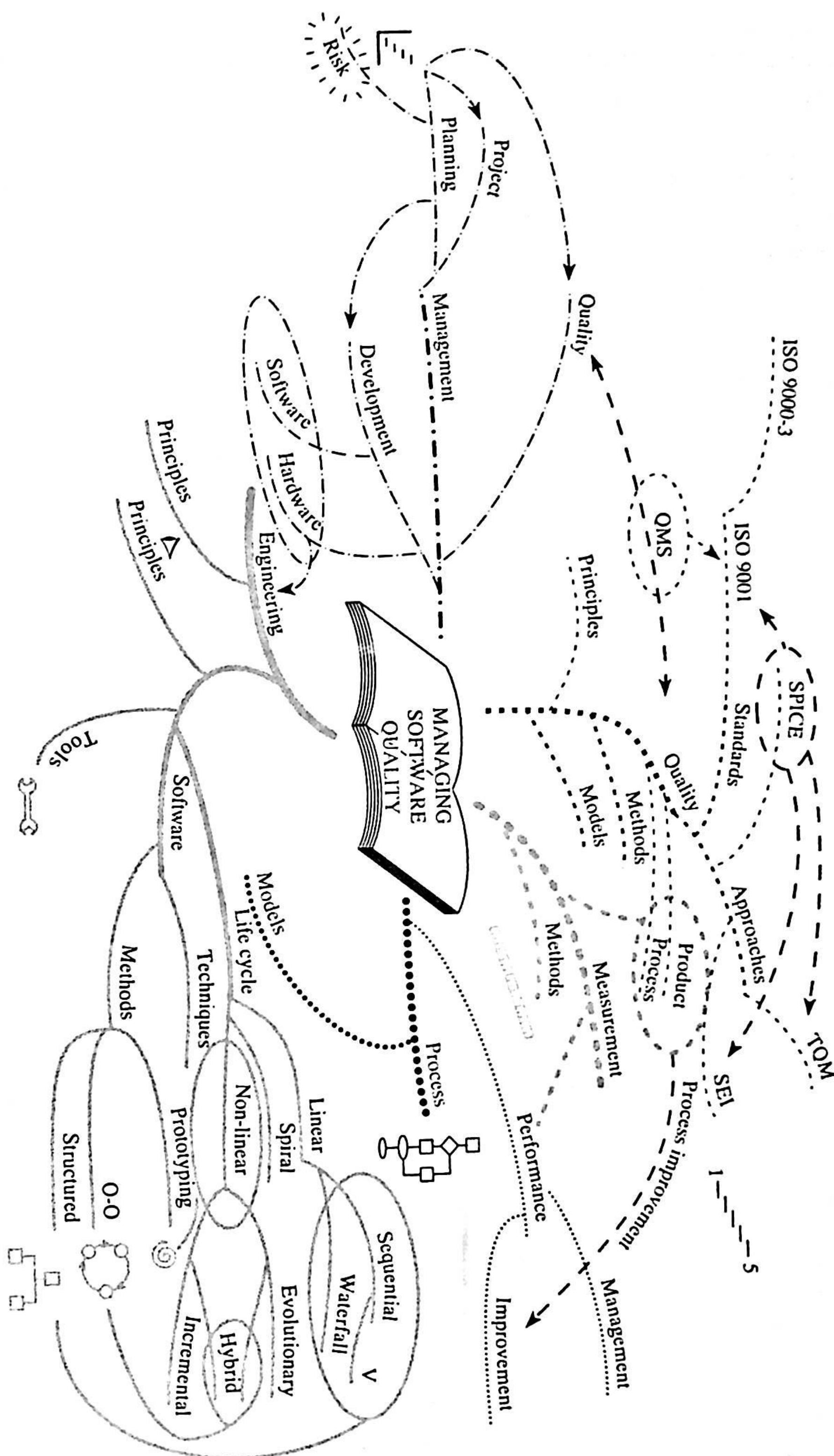
ממרחק מתאים בו מיטשטשים הפרטים, והקשרים משתלטים על התמונה. הסתכלות מרחוק על ציור מעניקה לנו את הרושם שאליו התכוון הצייר; גישה דומה מסייעת לנו לקבל מראה הוליסטי של מה שנראה כמצב מורכב ומבלבל.

מפות תפיסה משתמשות בדרך כלל בצבעים, בצורות ובדמויות, כדי להביע רעיונות וקשרים. בספר זה אנו מוגבלים לייצוג בשחור ולבן, ולכן השתמשנו בקווים מסגנונות שונים כתחליף לצבעים. דבר זה מקטין ללא ספק את כוח המשיכה ואת היכולת המרשימה של מפת התפיסה בכללותה. אנו מקווים שהסגנונות השונים של הקווים יהיה בהם כדי להבליט את צירופי הרעיונות השונים שבמפות.

לסגנונות הקווים אין משמעויות מוגדרות. הם משמשים רק לזיהוי מה שנמצא בתוך או מחוץ לצירופי רעיונות מסוימים. עם זאת, ניסינו להשתמש בסגנונות-קו דומים כדי לשקף צירופים דומים במפות התפיסה השונות. נקווה שתהנו מקריאת המפות, מהפרשנות ומההרחבה שלהן, וכי נוהל זה יפתח לפניכם מראות חדשים בנוף איכות התוכנה.

מפות תפיסה הן שעשוע של ממש וגם יכולות להיות מקור להתבוננות מעמיקה במערכות ובפעולות שעוסקים בהן. ההשראה למפות התפיסה שבספר זה מקורה בספרו של טוני בוזאן 'ספר מפת התפיסה' (The Mind Map Book - Buzan, 1990). אלו מפות פשוטות ביותר, וללא ספק ניתן לשפרן.

ועוד על מפות התפיסה: בשרטוט מפות תפיסה (mind maps) משתמשים בדרך כלל בצבעים, צורות, תמונות ותבניות שונות כדי לבטא רעיונות או יחסים. בספר זה השתמשנו בצבע אחד, שחור, ולכן ניצלנו קווים בסגנונות שונים, כדי להחליף את אפשרויות הצבע. צריך לשים לב לכך שסגנון קו מסוים מציין קבוצה של פעולות, או נושאים בעלי מכנה משותף.



תרשים 1 מפת התפיסה של הספר

חלק 1

מבוא להנדסת איכות תוכנה

מטרת חלק 1 לספק מבוא לעקרונות הבסיסיים של הנדסת איכות תוכנה ולזהות את הרעיונות העיקריים שאנו דנים בהם בספר זה.

חלק 1 מחולק לשני פרקים:

◊ פרק 1: הנדסת איכות התוכנה

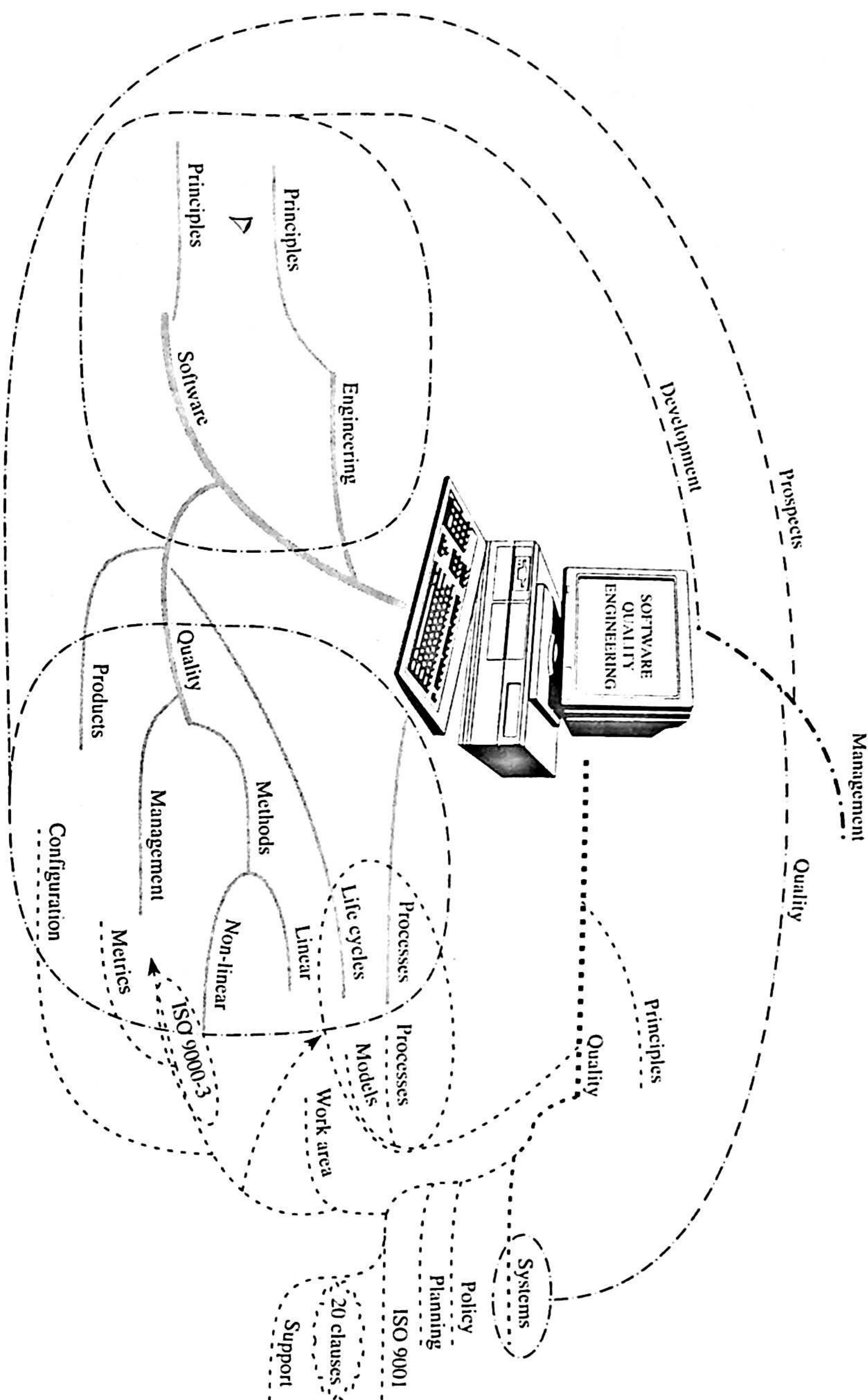
◊ פרק 2: מערכות לניהול איכות - מה מטרתן?

בפרק 1 נחקור משמעות המונחים 'תוכנה' ו'הנדסה' לפני שנצרפם יחד כדי לאפיין את הדיסציפלינות של 'הנדסת תוכנה'. אחר כך ננקוט בגישה דומה לגבי המונח 'איכות' ונחקור כיצד הוא מיושם לגבי התוכנה, כדי לאפיין את 'איכות התוכנה'. ולבסוף נצרף את כל הקצוות כדי לזהות מספר אלמנטים עיקריים של הנדסת התוכנה ושל איכות התוכנה, שיש לשקול במסגרת דיסציפלינה לניהול איכות תוכנה. לתוצאה זו נקרא בשם 'הנדסת איכות תוכנה'.

בפרק 2 נתבונן בצורה כוללת יותר בניהול האיכות, כדי לזהות את מרכיבי מערכת לניהול האיכות וכיצד משתלבים בה הרכיבים השונים זה בזה. נציג את דרישות תקן ISO 9001 ואת הקווים המנחים של תקן ISO 9000-3 כדי לראות כיצד הם מגלמים את עקרונות ניהול האיכות. מבוא קצר זה למסמכי ISO משמש גם כהפנייה לעיון נוסף בקווים המנחים שבפרקים המאוחרים יותר.

לכשתסיים חלק זה של הספר תהיה לך תפיסה מוצקה למדי של היסודות שבבסיסם של הנדסת תוכנה וניהול האיכות, כדי להבין ולהפיק תועלת מהטיפול בנושאי הנדסת איכות התוכנה שנציג בחלקי הספר האחרים.

רעיונות המפתח שמוצגים בחלק 1 מתוארים בתרשים ח.1, המציג דוגמה למפת תפיסה כללית.



תרשים ח.1 דוגמה למפת תפיסה כללית

פרק 1

הנדסת איכות תוכנה

מבוא

הנדסת איכות תוכנה היא המזיגה הנבונה של העקרונות ההנדסיים, של טכניקות התוכנה ושל שיטות האיכות, המפיקה בעקביות מוצרי איכות בעלויות ובלוחות זמנים סבירים. מציאת היחסים המדויקים של מרכיבי המזיגה היתה מאז ומתמיד יעד מבוקש של מהנדסי התוכנה ושל המשתמשים במוצרי תוכנה. אם יש ברצוננו להפליג במסע חיפוש אחר יעד זה, עלינו להבין שני דברים חשובים: כיצד נבחין ביעד לכשנגיע אליו, ומהו הכיוון בו נפתח בחיפוש זה? הגישה שננקוט לגבי שאלות אלו היא שתגדיר את הערכים שאנו עוסקים בהם, ותיתן לנו מפת דרכים עבור המסע - ועבור ספר זה.

תוכנה, איכות והנדסה

הצירוף של שלוש מילים אלו - **תוכנה, איכות והנדסה** - רווי כולו באי הבנות פוטנציאליות. מילים אלו, על אף היותן מושרשות כבר בשפת היומיום שלנו, מעוררות אוסף רחב ומגוון מאוד של רעיונות, ואף פעם אין ביטחון באיזה מהפירושים הרבים אנו משתמשים כאשר אנו מנסים לתקשר עם הזולת, גם אם הוא חבר למקצוע.

אין כוונתנו ליצור בספר זה אוצר מילים בעלות מובן חד-משמעי. עם זאת, יש צורך להגיע להסכמה בדבר תמצית הרעיונות המסתתרים מאחרי שלוש מילות המפתח 'תוכנה', 'איכות' ו'הנדסה'. נקדיש מאמץ מסוים להסברת מילים אלו, כדי שנוכל לבנות סביבן מסגרת רעיונית. נעשה זאת בגישה שתמחיש באמצעות דוגמאות והשוואות, את שיש ברצוננו להביע.

התוכנה מהי

מבין שלוש המילים הנזכרות, השם **תוכנה** הוא הפחות צפוי לאי הבנות. קל להגדיר אותו במילים אלו, למשל: 'הדבר שמקנה למערכת מחשב את ההתנהגות האופיינית לה'. זהו צירוף מילים חופשי מדי מכדי שיוכל לשמש כהגדרה, אך כוללני במידה מספקת שמאפשר גיוון מסוים. ניסוח זה למשל, יכול לאפשר את הכללת השימוש

בחבילות סטנדרטיות, בצד פיתוח יישומים מורכבים המשתמשים בארכיטקטורות מחשבים המיועדות למטרות ספציפיות מאוד.

הבעיות הנלוות לתוכנה, כגון אמינות נמוכה, חוסר גמישות ועלויות פיתוח חריגות, חלות על כל הרמות ועל כל סוגי התוכנה. בין אם נתחיל לעצב תוכנה מנקודת האפס, נתחזק תוכנה קיימת, נתאים חבילות תוכנה לצרכינו, נבחר ונתקין חבילות סטנדרטיות, או פשוט ננסה לנצל בדרכים חדשות את מערכות המחשבים הקיימות מבלי לכתוב תוכניות כלל. בכל המצבים האלה נִמְצא ב'עולם התוכנה' ונהיה כפופים לתרבות הייחודית לה. בהקשרים שבספר זה, כל פעילות המיועדת להשיג תוצאה מסוימת על ידי השימוש במחשב כחלק מתהליך כלשהו, נחשבת כפעילות של תוכנה.

האיכות מהי

ידועים הקשיים הרבים שבהגדרת **איכות**. בדומה ליופי, האיכות נמצאת בדרך כלל 'בעיני המתבונן', ובדרך כלל אנו מתפשרים על אמירות מפוקפקות, כגון 'כאשר אראה אותה, אבחין בה'. אמירות מסוג זה עשויות להכיל מידת מה של אמת, אך אין בהן כדי לספק קו מנחה למי שמנסה ליצור מוצרים שיענו דרך קבע על ציפיותינו.

גם אם יש פגמים בנסיונות שלנו לאפיין את האיכות, חייבת להיות לנו מטרה כלשהי שנוכל לשאוף אליה. בסביבה עסקית, האיכות מתבטאת במשמעות פורמלית שאפשר להביעה בקבוצה מגובשת של דרישות שיסופקו במועד מאוחר יותר, ושאפשר יהיה למדוד אותן, כך שנוכל להיות בטוחים באמת, שאכן סופקו לנו הדרישות שהצגנו.

> ***בעולם המושלם, כולם יראו את אותו דבר בעת שיחשבו על 'איכות'; הם יראו את אותו הדבר לפני חתימת החוזה וגם לאחר אספקת נשוא החוזה.***
>
> ***קאמרון לאו (SQM, 1993).***

אם כן, כיצד נגדיר את 'איכות'? נצטט רק שלוש מתוך ההגדרות היותר 'מכובדות', שהושמעו בעבר, כגון 'התאמה למטרה', 'תאימות לדרישות' ו'דרגת המצוינות'. אף לא אחת מהן נכונה באופן בלעדי. אף אחת מהן גם אינה הולמת בשלמות את הנושא, וכולן יחד אקדמיות במידה רבה.

בפועל, אם ברצוננו לשרוד בעולם העסקים, כל מה שחשוב לנו, היא השאלה אם הלקוח אמנם **שבע רצון**, או אפילו מאושר מהתוצאות. עלינו לוודא שאנו מתמקדים כל הזמן במשאלות ובצרכים של הלקוח, בין אם הוגדרו במלואם ובבהירות במפרט מתאים של הצרכים, ובין אם לאו. גישה שיטתית לאספקת איכות מחייבת בהכרח יצירת הגדרה מוסכמת של הדרישות, ואחר כך, אישוש העובדה שהמוצרים אכן תואמים דרישות אלו.

בהנחה שאין ביכולתנו להגדיר כבר מלכתחילה בצורה ברורה וחד-משמעית את מה שאנו רוצים, עלינו לוודא יכולת של מתן ביטוי לשינויים ולהתפתחויות כדי שנוכל להוסיף וללמוד את הצרכים, תוך כדי התקדמות התִיכון, ולשנות את מוצרינו כפי שיידרש, כתוצאה מהבנתנו המחודשת את הצרכים האלה.

בספר זה, המונח 'איכות' ישמש במשמעות של השגת **'שביעות הרצון של הלקוח'**. דבר זה פירושו אספקת מוצרים שיענו במועד מוסכם על קבוצה מוגדרת ומוסכמת של דרישות. תוך מתן ביטוי להתפתחות מתמשכת של דרישות אלו. הדרישות עשויות לכלול גם צרכים פונקציונליים (מה חייב המוצר לבצע) וגם צרכים לא-פונקציונליים (כיצד חייב המוצר להתנהג).

מה הקשר של ההנדסה לכאן

לא לכל תוכנה יש קשר להנדסה, כשם שגם המושג 'איכות' אינו שמור למהנדסים בלבד. מדוע אם כן, מוקדשים כל כך הרבה ספרים לנושא 'הנדסת תוכנה' ומדוע מתוארות טכניקות האיכות לעיתים כה תכופות בהקשר של ההנדסה? הסיבה היא שהדיסציפלינה (discipline) של ההנדסה היא כה מוצלחת בתחומה, עד שגם אחרים ניסו ללכוד את תמצית הדיסציפלינה הזו ולהשתמש בה כמודל לפיתוח שיטות, נהלים ומשמעת עבודה, שיהיו מוצלחים גם בתחומי התמחות אחרים.

מה שאנו מבינים היום תחת השם דיסציפלינה הנדסית, הלך והתפתח במרוצת מאות השנים, והיום אפשר לראות בו מכשיר אמין ועקבי לבניית כלים ומערכות, הבאים לענות על צורכי אנוש. יש תחומים, כגון בניין אוניות והנדסה אזרחית, שהפכו לשגרתיים יחסית, אף שלעיתים תכופות הם עוסקים בבעיות קשות ומורכבות מאוד.

המהנדסים פיתחו מספר עקרונות בסיסיים המשמשים אותם נאמנה כבר דורות רבים. כדאי לנו להקדיש מאמץ מסוים ולהתבונן באחדים מעקרונות אלה.

שמונה עקרונות הנדסיים חשובים

1. **תכנן לפני שאתה בונה**

המהנדסים יודעים שמשתלם לתכנן את הדרך בה אתה מתכונן לבנות דבר מה, לפני שאתה חותך את החומר ממנו תבנה את המוצר; אחרת, אתה צפוי לבזבוז רב בשל התחלות מוטעות, ואיכות המוצר תשקף בוודאי את הגישה האקראית של הבנייה.

התפיסה של הגדרת תהליך הפיתוח ותיעודו מבוססת על גישה זו. המתודולוגיה של התִיכון מספקת מבנה של תכנון מפורט יחד עם **מוצרים מוגמרים** (deliverables) מוגדרים היטב בכל שלב, וזיהוי הנקודות בהן יש לקיים ביקורות, כדי לוודא שהתִיכון מתבצע בהתאם לתוכנית.

2. **וודא שהחלקים יתאימו זה לזה**

המהנדסים יודעים שהדרך היחידה בה ניתן להשיג תאימות בין חלקים המיוצרים בזמנים ומקומות שונים, היא על ידי הסכמה על דרך סטנדרטית לתיאור החלקים האלה, ועל ידי שימוש בכלים ובמכונות שיכולים לפעול במידת הדיוק הנדרשת. כך יהיה ביכולתם לוודא שהחלקים יתאימו זה לזה כאשר יצרפו אותם יחד.

3. **תכנן את המבחנים לפני הבנייה**

המהנדס המגדיר את החלקים שייוצרו עבור העבודה המסוימת חייב גם להחליט על הדרך שבה ייבדקו וייבחנו החלקים המיוצרים; אחרת, קיימת סכנה שהבדלים בייצור יתבטאו בחלקים שאינם מתאימים זה לזה, או שדרך פעולתם לא תהיה תקינה. דרישות אלו של הבדיקות והניסויים חייבות להיות מוגדרות מלכתחילה בצורה שתבהיר במפורש ומראש מהו הדבר שנדרש, תאפשר את השימוש בדרישות לבקרה של תהליך הייצור, ותמנע את הבעיות שהן אמורות לחשוף לאחר גמר הייצור.

4. **בדוק את התִיכון לפני כניסה למחויבות**

ייצור על פי שרטוטים יש בו כדי לוודא את העקביות, אך לא את הנכונות, אלא אם כן נבדקו תחילה השרטוטים עצמם ונמצאו נכונים. מסיבה זו, שרטוטי התִיכון הראשוני מאומתים בדרך כלל על ידי המעצב, כדי לוודא שלא השתרבבו שגיאות תוך כדי פעולת השרטוט עצמה, וכדי לערוך בדיקה סופית שמטרתה לוודא שהתִיכון עצמו הוא אמנם נכון.

5. **דאג לקיים בקרה מתמדת על הכנסת שינויים בתִיכון**

משעה שהשרטוטים מועברים אל פסי הייצור, חשוב לקיים בקרה קפדנית על כל השינויים המתבצעים בשרטוטים. כל שינוי שלא אושר על ידי המעצב עלול לפגום בתקפות של התִיכון כולו, ובוודאי גם לגרום לאובדן מחזור ייצור שלם. כל שינוי שאינו חייב להיעשות על ידי המעצב עצמו, צריך להישקל בזהירות ולהיות נתון לבקרה שתוודא את צמצום ההשפעה על הייצור עד למינימום. כל השינויים חייבים להשתקף גם בדרישות הבדיקות והניסויים, משום שאחרת, הניסויים לא יהיו אפקטיביים.

6. **וודא שאתה אכן מספק את האיכות הנכונה**

משעה שמתחילים בייצור, חיוני לוודא שכל הרכיבים המיוצרים (או לפחות מדגם של מוצרים אלה) תואמים את המפרטים, וכי תהליך הייצור אינו תורם באופן שיטתי לשגיאות ייצור כלשהן. **בקרת האיכות** (quality control) **ואבטחת האיכות** (quality assurance) מספקות שירותים קריטיים אלה.

7. **למד משגיאותיך והשתפר בפעם הבאה**

השימוש במידע סטטיסטי המתייחס למוצרים ולתהליכי הייצור, מאפשר את צמצום הבזבוז ומשפר את התהליכים להגדלת היעילות ושביעות הרצון של הלקוחות. בארגונים רבים, איסוף נתונים מקווי הייצור, ממחלקות התכנון והתִיכון, משירותי התמיכה ללקוחות ומדומיהם, מהווה חלק מתהליך הפעילות השגרתית. רבות מההחלטות המסחריות מותנות באספקה במועד הנכון של נתונים מדויקים בדבר איכות המוצר והתהליכים.

8. **דע לאן אתה הולך**

אי אפשר להגזים בתיאור החשיבות והמשמעות של פעולה על פי אסטרטגיה כוללת, המבוססת על יעדים ברורים. חברות מצליחות הן חברות ששמות להן למטרה להשיג יעדים ספציפיים ורותמות את משאביהן בצורה אפקטיבית להשגת

יעדים אלה. האסטרטגיה הכוללת הזו אמורה להכיל הגדרה ברורה של יחס החברה ומחויבותה לאיכות. דבר זה חיוני לניהול הטוב, משום שהייצור של מוצרי איכות ייראה תמיד כאילו הוא יקר יותר, ורק מחויבות אמיתית לאיכות תמנע קיצוץ בעלויות שיהיה גם מלווה בפגיעה באיכות. אם האיכות היא אמנם מרכיב יסודי בתדמית החברה, יש לשקוד על כך שתהיה לנחלת הכלל ותהנה מפרופיל גבוה בחברה.

הנדסת תוכנה

האם נוכל לרתום עקרונות מנחים אלה שמתחום ההנדסה בצורה שתוכל לשרת אותנו גם בתחום פיתוח התוכנה? **הנדסת התוכנה** (Software enjineery) היא לפחות ניסיון בכיוון זה.

ההגדרה המוקדמת ביותר להנדסת התוכנה ניתנה כנראה על ידי פרופסור פריץ באואר בשנת 1969 :

מיסוד ושימוש בעקרונות סבירים של ההנדסה כדי להשיג בדרך כלכלית תוכנה, שהיא אמינה ופועלת ביעילות במחשבים אמיתיים.

הדיסציפלינה ההנדסית סיפקה בשעתה מודל אפשרי יחיד שעליו ניתן היה לבסס את הגישה המקצועית לפיתוח תוכנה, והכותרת **'הנדסת תוכנה'** העניקה מידה מסוימת של מכובדות למה שהיה באותה עת אוסף אקראי של רעיונות השונים זה מזה, המדברים על הדרך בה יש לבצע את פיתוח התוכנה. בתקופה שחלפה מאז, הלכה ובשלה הדיסציפלינה החדשה ויישומים חדשים הופיעו ועלו במגוון רחב של תחומים. במקביל, גדל גם מספר מפתחי היישומים בעלי רקע שאינו דווקא מתחום ההנדסה, ואלה עסקו בבניית מערכות שהקשר שלהן אל ההנדסה היה מצומצם, או כלל לא קיים. כתוצאה מכך, ייתכן שהתואר 'הנדסת תוכנה' פגע אולי במספר לא מבוטל של מפתחי תוכנה מקצועיים, אך עם זאת הפך להיות חלק בלתי נפרד מתרבות התוכנה.

ההגדרה של פרופסור באואר אינה בשום מקרה ההגדרה היחידה שהוצעה לציבור, אך היא שימשה כנקודת מוצא נוחה, ואין ספק שהיא מכילה מספר אלמנטים עיקריים שלא הוצבו יחד קודם לכן. השימוש בביטוי **'עקרונות הנדסה סבירים'** (sound engineering principles) בהגדרה דלעיל, מדגיש את הכוונה לפעול לפי הדמייה של הגישה ההנדסית, אם כי לא ברור כיצד יושג הדבר בפועל. יש מקום לכך שנעניק לביטוי זה את המשמעות הבאה: מיסוד דיסציפלינה הנדסית סבירה ומוסכמת, בצירוף עם קבוצת רעיונות בסיסיים על האופי של פיתוח התוכנה. ניסוח זה מרמז למשל, על הסכמה בדבר מודלים של **מחזור חיים** (life cycle models) ומתודולוגיות מתאימות. מילות מפתח אחרות הכלולות בהגדרה - משקית, אמינה, ופועלת ביעילות - ממחישות את הדאגה להיבטים הבלתי-פונקציונליים של מוצרי התוכנה וגם לדרישות הפונקציונליות. בהתחשב במהירויות ובעוצמות החישוב הכמעט בלתי מוגבלות של

ימינו, עשויה היעילות שלא להיות נושא לדאגה כה מקיפה, אך העיקרון של זיהוי **תכונות** (attributes) רצויות ובניית מוצרים שיספקו אותן, ממשיך להיות עיקרון בעל ערך רב.

מה שהגדרה זו אינה מספקת, הוא התהליך באמצעותו ניתן להשיג את התכונות הרצויות. המשימה שלנו בספר זה היא להגדיר את תהליך הפיתוח בצורה שניתן יהיה ליישמו בצורה שימושית בארגונים גדולים וקטנים, ויאפשר לנו להפיק מוצרים בעלי איכות מוגדרת העונים על צורכי הלקוחות.

נקודה נוספת מתוך ההגדרה, שחשוב לציינה, היא הדגש על השימוש במילים 'כדי להפיק' ולא במילים 'לעצב, לתכנת, או לבנות מערכות'. הנדסת תוכנה פירושה יותר מאשר רק כישורים טכניים. ההגדרה מכירה בכך שהנדסת תוכנה היא דיסציפלינה שחייבת להקיף את כל הפעילויות החיוניות, המוודאות שהלקוחות יקבלו מערכות שעונות על צורכיהם. עיון חוזר בעקרונות ההנדסיים שתוארו לעיל יסייע לנו לזהות אחדות מהפעילויות החיוניות.

שמונת העקרונות של הנדסת התוכנה

1. **הגדר את מחזור החיים של פיתוח תוכנה (תכנן לפני שאתה בונה)**

מהנדסי תוכנה זקוקים להגדרה של **מחזור החיים של הפיתוח** (development life cycle) כדי שיוכלו לדעת באיזה סדר לבצע את הדברים, וכדי שיוכלו לתקשר עם הזולת בכל הנוגע לתהליך ולהכניס בו התאמות במקרה הצורך. מתודולוגיית פיתוח מעדנת את התהליך המבוסס על מחזור חיים, כדי לסייע בתכנון וכדי לספק מסגרת לשיטות תִיכון ו**לשיטות רישום נתונים**.

2. **תן פירוט של הממשקים (וודא שהחלקים יתאימו זה לזה)**

מהנדסי תוכנה זקוקים להגדרות מדויקות של מה שצריך לבנות, הגדרות שתינתנה בצורת מפרטים. דבר זה חשוב במיוחד כאשר רכיבי תוכנה חייבים לפעול ביחד. הגדרה זהירה של הממשקים שבין רכיבי התוכנה יש בה כדי לוודא שהרכיבים יתאימו זה לזה בצורה נכונה, וגם כדי לאפשר את בניית הרכיבים על ידי עובדים שונים בצוות. כלים התומכים בתהליך הבנייה מסייעים לצמצום הסיכויים לטעויות בממשקים.

3. **תכנן מראש את הבדיקות (תכנן את המבחנים לפני הבנייה)**

תכנון הבדיקות (tests) ומפרטי הניסוי קובעים את הדרישות לביקורת עוד לפני הייצור. על ידי כך ניתן לוודא תשתית אובייקטיבית לבדיקות, שתהיה מבוססת על הדרישות ותספק בסיס לביקורות תוך כדי התהליך, ובכלל זה בדיקות של רכיבי התִיכון ההולך ומתגבש.

4. **בחן מחדש את התִיכון (בדוק את התִיכון לפני כניסה למחויבות)**

בחינה חוזרת של התִיכון (design) היא שיטה לאימות התִיכון שיש בה כדי לוודא עוד לפני התחלת 'הייצור', שלא השתרבבו שגיאות לתוך התִיכון.

5. **נהל את התצורה (דאג לבקרה מתמדת על הכנסת שינויים בתִיכון)**

לגבי מהנדס מערכות, **ניהול התצורה** (configuration) הוא המקבילה של בקרת השרטוט; עם זאת, בתחום התוכנה, הקלות היחסית בה ניתן לשנות את המוצרים, מגבירה עוד יותר את החשיבות של שימת דגש על משמעת ניהול השינויים.

6. **נהל את האיכות (וודא שאתה אכן מספק את האיכות הנכונה)**

האופי של רוב מוצרי התוכנה, שהם בלתי נראים לעין ובלתי מוחשיים, מקנה חשיבות רבה ביותר למתן שירותי בקרת איכות ואבטחת האיכות, ומקשה על מתן שירותים אלה. בדיסציפלינה שמבוססת במידה רבה ביותר על שיטה מכוונת - תִיכון (design-oriented), ובעלת שלב ייצור טריוויאלי, אימות התִיכון חשוב במיוחד והביקורת של המוצרים הפיסיים הנמסרים ללקוח (deliverables) אין בה תועלת רבה. הנדסת תוכנה עוסקת הרבה יותר בפונקציה (כלומר בביצועים) מאשר בצורה.

7. **מדוד את המוצרים והתהליכים (למד משגיאותיך והשתפר בפעם הבאה)**

השיטה למדידת התוכנה נמצאת עדיין בחיתוליה, אך יש צורך לפתח אותה, כדי לספק תמיכה למנהלים האחראים להעשרת תהליך פיתוח התוכנה.

8. **הגדר את מדיניות האיכות (דע לאן אתה הולך)**

לגבי מפתחי התוכנה, היכולת לספק מוצרים בעלי איכות הנראית לעין, היא עתה משימה בעלת חשיבות ראשונה במעלה. האמון במוצרי תוכנה נמצא היום בשפל, וזאת דווקא בשעה בה מוצרי התוכנה משפיעים במידה גדלה והולכת על חיי היומיום של הציבור הרחב. תחומי יישום חדשים צצים ועולים מדי יום. עימם גדל גם הצורך בהקמת מנגנון אמין, שבאמצעותו ניתן יהיה לנהל את איכות המוצר, כדי לעמוד באתגר של הצורך באמון רב יותר.

דגש זה על **התהליך** של פיתוח התוכנה חודר בהדרגה לתודעה של תעשיית התוכנה. היו כבר בעבר התחלות שנכשלו או שהסתיימו במבוי סתום, אך החיפוש אחר תוכנה איכותית בעלת רמה, הוליך את מהנדסי התוכנה להתמקד פחות במחשבים ויותר בטכנולוגיה שבאמצעותה נעשה שימוש במחשבים. לטכנולוגיה זו מספר צרכים בסיסיים:

- אמצעים שיאפשרו את **התקשורת** עם המחשבים באמצעות שפות התכנות;
- **שיטות וכלים לתִיכון**, והתודעה שמקום התכנות אינו בתחילת הדרך, אלא דווקא אי-שם קרוב לסוף;
- ההגדרה של המונח **מחזורי חיים של הפיתוח** שממקמת את כל הפעילויות הרלבנטיות, מהתפיסה הראשונית עד לסיום הסופי, בסדר לוגי.
- זיהוי **המתודולוגיות של הפיתוח** כדרך שתשמש להגדרת השלבים ושל הפריטים **הנמסרים** (deliverables), שיש לאמצה במסגרת מחזור חיים נתון;
- ההכרה בכך שמערכות תוכנה חיות בתוך **סביבה** המשפיעה על הדרישות הספציפיות של המשתמש, ועל התרבות בה ייושמו וייעשה בהן שימוש.

הנדסת התוכנה מספקת את הבסיס לתהליך פיתוח התוכנה. היא מתמקדת באספקת מוצרים שעונים על דרישות מוגדרות, כולל דרישות לאיכות, והיא מביאה בחשבון את מחזור החיים השלם של הפיתוח.

אך כיצד נגדיר את הדרישות לאיכות התוכנה?

איכות תוכנה

אם האיכות משמעותה שביעות רצון של הלקוח, כיצד נוודא שאנו אמנם משיגים אותה? לקוחות בסופו של דבר הם רק בני אנוש, ומסוגלים לשנות את דעתם, או אפילו לא לדעת מה הם בדיוק רוצים. מהר מאוד ניווכח לדעת ששינויים הם חלק מכל תהליך פיתוח תוכנה, ואם ננהג בחכמה, נלמד להיענות לשינויים אלה. יחד עם זאת, יהיה זה מסוכן מאוד לפתוח בפרויקט כלשהו, מבלי שתהיה לפנינו נקודת מוצא ברורה. כבר מההתחלה חייב להיות לנו רישום של מה שהלקוח חושב שהוא רוצה בשלב זה, גם אם רצון זה אינו מלא, אינו נכון או אפילו בלתי ניתן להשגה; שאם לא כן, לא יהיה לנו קו מוצא בסיסי, שבו צריך יהיה להכניס את השינויים במועד מאוחר יותר. הגדרת דרישות הלקוח, כולל התכונות המופיעות תחת הכותרת 'איכות', היא המוצר החיוני הראשון של כל פרויקט תוכנה.

הגדרת דרישות

מה צריכות הדרישות לכלול? נעשו מספר נסיונות להגדיר קבוצה אובייקטיבית של קריטריונים לאיכות, אך אף אחד מאלה לא הביא פתרון קבע לבעיה. מבחינה מסוימת, אין גם סיכוי למצוא פתרון כזה, משום שאין בפתרון זה כדי לוודא שהלקוח אכן התייחס כבר במועד מסוים להיבטים שהם קריטיים ליישום מסוים.

> ***'מוצר איכות' מבצע את רוב הדברים שאנו רוצים שיבצע, וגם נוכל להשלים עם הפגמים שקיימים בו.***
>
> ***פאול הרצליך (SQM, 1993)***

הקריטיות היא הציר המרכזי של נושא האיכות. הלקוחות עשויים שלא להבין את כל המשמעויות של הדרישות העסקיות שלהם, אבל עליהם להחליט מהם האלמנטים שחיוניותם מוחלטת. מהם הגורמים הקריטיים להצלחתו של הפרויקט הזה? אם נצליח ללכוד את הנתונים האלה בצורה כמותית - למשל, בצורת קבוצה של גורמי הצלחה קריטיים - יעמוד לרשותנו בסיס ממשי שנוכל לשפוט באמצעותו אם אכן אנו בונים את מה שהלקוח באמת מעוניין בו.

השגת איכות התוכנה

נאמר כבר שהשגת האיכות פירושה 'יצירת תוכנה שלא תחזור אליך, עבור לקוחות שכן יחזרו אליך', ובכל הנוגע למטרות ספר זה, הגדרה זו יכולה בהחלט לענות על הצרכים. כשלרשותנו הגדרה פרגמטית של האיכות, המלווה בגישה מציאותית אל דרישות

האיכות, כל מה שנדרש לנו הוא תהליך שבאמצעותו ניתן להשיג את דרישות האיכות האלו בצורה אמינה ועקבית. כיצד ייראה התהליך הזה?

הדיון שלעיל מרמז על כך שהתהליך שבאמצעותו אנו מספקים את האיכות צריך להיות מבוסס על הדיסציפלינה של הנדסת התוכנה, אך האם עלינו להמשיך גם הלאה בתהליך זה? האם יש ביכולתנו להגדיר תהליך כוללני שיפיק איכות בצורה שיטתית, ושיגלם בתוכו את שמונת העקרונות ההנדסיים בצורה שלא תרחיק ממנו את עובד הפיתות שאינו בעל רקע הנדסי? בפרק 4 נדון ביתר הרחבה בתהליך הפיתוח הבסיסי.

תהליכי איכות

מהו תהליך איכות? לצורך הדיון שלנו, **תהליך איכות** (quality process) הוא תהליך המכיל שלושה מרכיבים:

- אנו יודעים מה נדרש מהתהליך;
- אנו יודעים כיצד הוא פועל;
- אנו יודעים כיצד לתקן או לשפר אותו.

התייחסנו כבר לקריטריון הראשון. ההתייחסות לקריטריון השני נדונה כבר חלקית כשעסקנו בטכניקות הנדסת תוכנה. עלינו לבחור שיטות ספציפיות ולתאר את הדרך בה הן פועלות, כדי שנוכל לשנותן ללא סיכון, במקרה שיתברר שהן אינן עונות בדיוק על הצרכים שלנו.

בדיון על עקרונות ההנדסה, נגענו בתחום **אבטחת האיכות** (quality assurance). רכיב זה הוא שמוסיף את המנגנון של תיקון-עצמי ושיפורים, ולכן עלינו להוסיף אותו לדיסציפלינת הנדסת התוכנה שלנו, כדי להגיע לשיטה של הנדסת איכות תוכנה. תחום זה לא נדון עדיין במלואו, ונשוב אליו ביתר פירוט בפרק 5.

לאחר צירוף כל שלושת המרכיבים האלה, אמורה לעמוד לרשותנו קבוצת תהליכי איכות שנצמדים לשמונת העקרונות ההנדסיים. הם ישקפו את הנוהג הטוב ביותר בענף התוכנה, ויספקו את האמצעים לתחזוקתם ולשיפורם העצמי. אוסף כזה של תהליכי איכות קרוי בשם **מערכת לניהול איכות** (quality management system).

תקני מערכת לניהול איכות, כדוגמת התקנים הכלולים במשפחת ISO 9000, מנסים ללכוד את הנוהג הטוב ביותר, בעיקר מתעשיית הייצור. אלה מספקים את המסגרת שבתוכה ניתן לשבץ דיסציפלינות כמו **הנדסת תוכנה** (software engineering).

תהליכי הנדסת תוכנה

מהו הדבר שאמנם מייחד את תהליך הנדסת התוכנה? מבחינות מסוימות אין זה אלא מקרה פרטי של תהליך הנדסי כלשהו, אלא שהוא כולל כמה היבטים ייחודיים, ועלינו לוודא שאנו מביאים אותם בחשבון במלואם, בשעה שאנו מתארים את התהליך ומבקשים לשלוט בו.

מהו הדבר שמייחד את הנדסת התוכנה

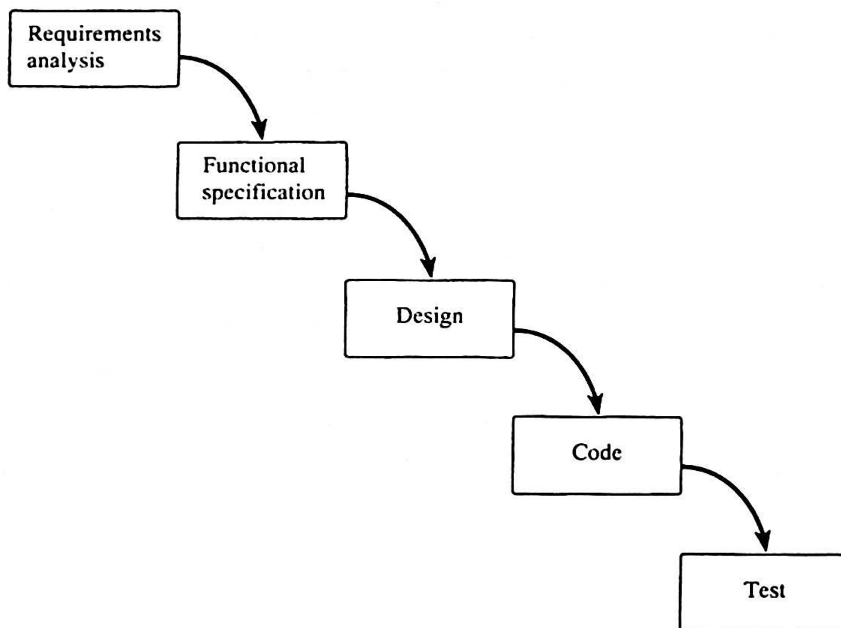
1. סביר להניח, שהמאפיין המשמעותי ביותר של פיתוח תוכנה, היא כמות האיטרציות הכלולות בתהליך. כל תיכון כולל בתוכו איטרציות, אלא שהנדסת התוכנה מתבצעת בדרך כלל בהקשר של מורכבות ואי-ודאות בדבר הדרישות האמיתיות. כתוצאה מגורמים אלה ומחוסר הבשלות היחסי של שיטות תִיכון התוכנה, חשופה פעילות התיכון לטעויות, והיא צפויה לכמות רבה ביותר של שינויים. לכן חייב תהליך התיכון להכיר במשמעות הנובעת מהאיטרציות.
2. לפיתוח התוכנה יש מוניטין של אי עמידה בתקציב ובלוח הזמנים. הסיבות לכך אינן פשוטות, אלא שהתחושה של ביצועים גרועים בכל הנוגע לניהול פרויקטים מחוללת לחץ להוספת שיפורים. תהליך הנדסת התוכנה חייב להיבנות בצורה שאפשר יהיה לבנות דיסציפלינה של ניהול פרויקט-תוכנה שתהיה מבוססת על התהליך ותוכל לנהלו ביעילות.
3. תהליך הנדסת התוכנה חייב להביא בחשבון את כל הפעילויות הכרוכות באספקה של תוכנה 'איכותית', החל בהעלאת הדרישות ההתחלתיות דרך האספקה ללקוח, קבלה על ידו, ומשם גם לתחזוקה. קרוב ל- 80 אחוז מההשקעה של המשאבים במוצר תוכנה מתבצעים רק לאחר הקבלה הראשונית של מוצר התוכנה על ידי הלקוח, ומשום כך עלינו להביא בחשבון את התהליך כולו ובשלמותו. נקודת המבט הזאת של תהליך הנדסת התוכנה נקראת בשם **מחזור חיים** (life cycle).

כתוצאה מהאמור לעיל, אנו יכולים לקבוע שהנדסת התוכנה מחייבת הגדרת מודל של מחזור חיים איטרטיבי, הכולל מידה מספקת של פירוט שתאפשר למנהלי פרויקטים לבנות וגם לבצע תוכניות מציאוּתיות.

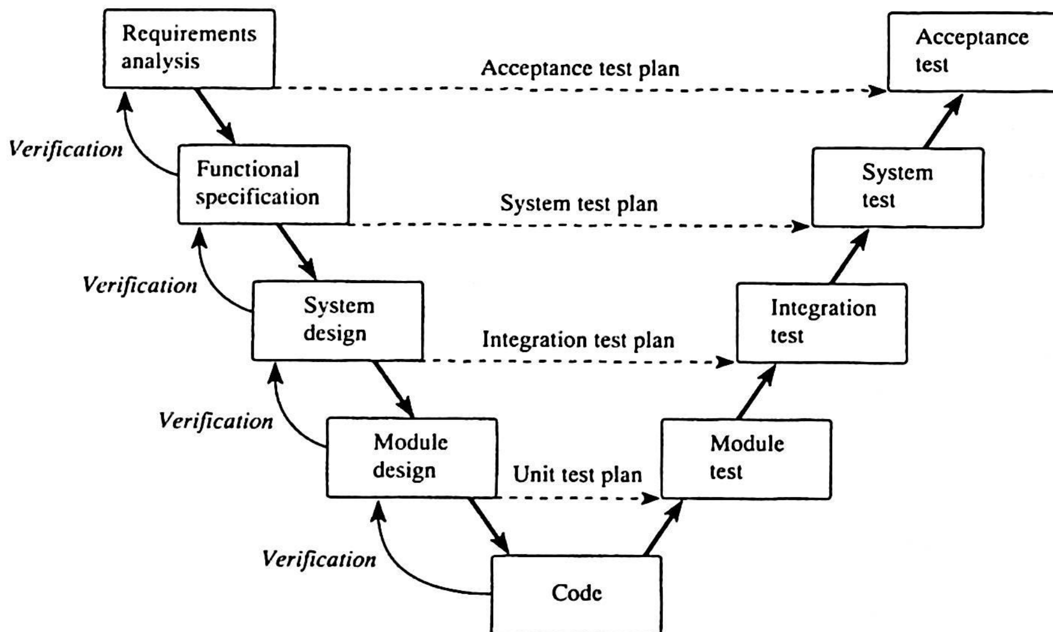
מודלים המבוססים על מחזור חיים

מודל המבוסס על מחזור חיים הוא בפשטות תיאור לתהליך שלם של פיתוח ואספקת תוכנה. מודלים של מחזור חיים מוצגים בדרך כלל בצורה גרפית. קיימים מודלים 'סטנדרטיים' אחדים שמשמשים לאפיון גישות שונות לנושא פיתוח התוכנה, ומאפשרים את הדיון ביתרונות ובמגבלות שלהם.

מודל מחזור החיים (life cycle model) המוקדם יותר וכנראה גם הידוע ביותר הוא **מודל-מפל-המים**, המוצג בתרשים 1.1. המגבלה הראשית של מפל המים היא בהיותו בעל אופי עוקב, סדרתי, שמשאיר את הבדיקות לשלב מאוחר בחיי הפרויקט, כאשר תיקון השגיאות יהיה יקר וגוזל זמן. שינויים שונים במודל מפל המים ניסו להדגיש את האופי האיטרטיבי של פיתוח התוכנה, אלא שהרעיון הבסיסי של פאזה אחת הזורמת לתוך הפאזה שלאחריה מרמז על רציפות ופעולות סדרתיות, או קוויות (לינאריות).



תרשים 1.1 מחזור החיים של מפל המים



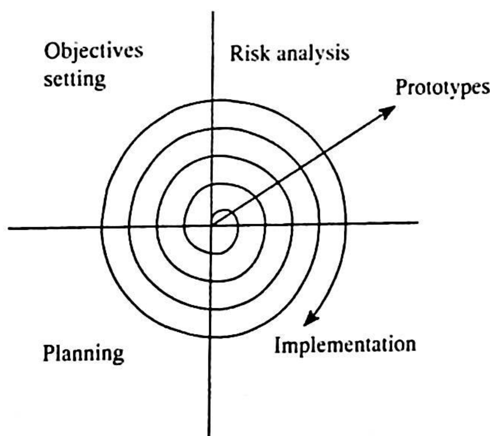
תרשים 1.2 מחזור החיים שבסגנון V

השינויים המשמעותיים ביותר שהונהגו במפל המים יצרו **מחזור חיים בסגנון V** (V life cycle). תרשים זה מציג מחדש את הרצף של מפל המים בצורה שמדגישה את הקשר שבין השלבים המוקדמים והמאוחרים של הפרויקט, ומפנה את תשומת הלב להזדמנות לתכנן ולהגדיר בדיקות בשלב מוקדם. תרשים 1.2 מציג את מחזור החיים שבסגנון V, המשמעויות של מודל זה נדונות בפרקים 3 ו- 4.

היתרונות שבגישה זו טמונים בחלקם בכך שמשיגים אובייקטיביות רבה יותר בבדיקות, ובחלקם בכך שמשתמשים בתהליך פיתוח הבדיקות, כדי למצוא ולסלק שגיאות בדרישות, במפרטים ובמסמכי התיכון של הדרג הגבוה. על אף היותו צעד נכבד קדימה אל מעבר למודל מפל המים, ממשיך מודל V להיות סדרתי, או קווי ביסודו, והדגש מושם בו על גישה של שלב אחר שלב.

מחזור החיים בסגנון V סייע בהדגשת המשמעות של האימות וגם של בדיקת התקפות במאמצים להשגת מוצרי איכות; מודל פשוט של **האימות** (verification) **ובדיקת התקפות** (validation) ניתן ליישם לגבי כל מוצר מוגמר, כלומר, לגבי הפלט המתקבל מכל שלב של מחזור החיים.

המודל הרדיקלי יותר שבין המודלים של מפל המים וסגנון V היה **מודל החילזון** (spiral model), שניסה לאפיין את האופי **האיטרטיבי** וההתפתחותי של פיתוח התוכנה. מעצם טבעו, מודל החילזון קשה יותר לניהול מאשר מחזורי החיים הקוויים, אך עם זאת אפשר לטעון שיש בו אפיון מדויק יותר של התהליך. מודל החילזון מוצג בתרשים 1.3. מודל זה זכה להתעניינות מחודשת עם התפתחותן המהירה של הגישות ליצירת **אבטיפוסים**, שיידונו בהרחבה רבה יותר בפרק 8.



תרשים 1.3 *מחזור החיים במודל החילזון (הספירלי)*

חשוב להדגיש בשלב זה שאין יתרון בשימוש במודל מסוים כלשהו של מחזור החיים. כל אחד מהמודלים ה'סטנדרטיים' הוא לא פחות 'טובי' מהמודל האחר, ורבים ממפתחי התוכנה פיתחו לעצמם מודלים שיש בהם מעט מאוד מהמשותף עם אחד

מהמודלים ה׳סטנדרטיים׳, ולעומת זאת, משקפים ביתר דיוק את גישתם לנושא הפיתות. השיקול הקובע הוא, אם מודל מחזור החיים הולם אמנם את צורכי הפרויקט שעוסקים בו.

המודלים הסטנדרטיים משמשים רק כדי להפנות את תשומת לבנו לאופי מחזור החיים וכדי למשוך אותנו לדו-שיח בנושא תהליך הפיתוח שלנו. המטרה היא שכל ארגון יכיר ויסכים על המודל או המודלים של מחזור החיים בהם הוא משתמש. יש מקום לכך שכל ארגון העוסק במגוון פעילויות של פיתוח תוכנה - כגון בית תוכנה - יגדיר לעצמו מודל של מחזור חיים עבור כל סוג של פעולת פיתוח שהוא עשוי לעסוק בה, ועבור כל פעולת פיתוח נתונה ישתמשו במודל ההולם אותה.

הנדסת איכות תוכנה

תהליכים ומוצרים

עד עכשיו התרכז הדיון בגישה אל איכות **מוכוונת-תהליך** (process oriented). ההנחה הבסיסית היתה שמוצרי איכות מופקים בתהליכי איכות. האם גישה זו נכונה בהכרח? המצב ההפוך תקף ללא ספק: העדר תהליכי איכות כלשהם צפוי לפגוע באיכות המוצר, לפחות בטווח הארוך. לעומת זאת, האם יש בכוחו של תהליך איכות להבטיח גם את איכות המוצר? בוודאי שלא.

מוצר שאינו עונה על צורכי הלקוח, לא יהיה בן שימוש, ואין זה משנה עד כמה היתה בנייתו טובה באמת. מבחינה מסוימת אפשר לומר שהמוצר חסר את האיכות. באותה מידה, מוצר שכן עונה על צורכי הלקוח, כן ייחשב למוצר איכות, מבלי להתחשב בדרך בה נבנה. רק למוצרי איכות יש ערך אמיתי.

שיטות מוכוונות-תהליך (Process-oriented methods) מייצגות את אחת הגישות האפשריות לפיתוח מוצרי איכות. ניתן לבנות תהליכי איכות שיתמכו ויעשירו את תהליכי הפיתוח הבסיסיים, ויספקו סביבה שמעודדת נהגי פיתוח טובים. אין בזה כדי להבטיח תִיכון טוב, אך הוא מצמצם את הסיכון של תִיכון טוב שנפגם על ידי נהגי פיתוח גרועים.

שיטות מוכוונות-מוצר (Product-oriented methods) עוסקות בצורה ישירה יותר בהשגת מוצרים טובים, אך שיטות אלו נוטות להיות **נטולות מסגרת** (open-ended) וקשות לניטור ולבקרה יעילה. הן אמנם מספקות נתיב מהיר וישיר המכוון למוצרי האיכות, אך חסרות בלמים ואיזונים המאפיינים את השיטות מוכוונות-התהליך.

שיטות מוכוונות-תהליך

מודלים של מחזור חיים כגון מפל המים ו-V מגלמים למעשה גישה ׳קווית׳. הם מתחילים בהצגת הדרישות, ממשיכים בהרחבת דרישות אלו למפרטים טכניים, ואחר לתִיכון שיושם, ייבדק, ויימסר ללקוח.

ההנחה הבסיסית שביסוד מודל ממין זה, היא שניתן להגדיר את הדרישות כבר בתחילת הפרויקט. הניסיון בהנדסת תוכנה אמיתית הראה שהנחה זו אינה תמיד תקפה. רק במקרים מעטים הושגה הגדרה נאותה של הדרישות, דבר הגורם לאיטרציות, שלעיתים תכופות לא הובאו בחשבון בתוכניות הפרויקט, וגורמות לכן לפיגורים. האיטרציות הבלתי צפויות גם מפעילות לעיתים תכופות לחצים על צוות הפיתוח, דבר הגורם לבעיות איכות.

תהליכי האיכות שתוארו עד כה, אם יישמו אותם ברצינות, ייקלו על ניהול הפרויקטים, ויגבירו את היתכנות אספקת מוצרי איכות. לגישות הקוויות יש **פתיחות** (susceptible) לגישת תהליך איכות. תשומת לב נאותה לתהליך הפיתוח תגרום להתגברות על חולשות הגישה הקווית, ובסופו של דבר להגדלת סבירות ההצלחה. התהליכים הקוויים נחקרים בפרקים 3, 4, ו- 5.

שיטות מוכוונות-מוצר

מחזור החיים החלזוני הוא 'בלתי-קווי', משום שהוא אינו מייצג גישה הדרגתית של שלב אחרי שלב, הניתנת לתכנון כבר מההתחלה. תהליך זה מאופיין על ידי שיטות 'בלתי-קוויות' שפעולת פיתוחן משתמשת בגישה התפתחותית של יצירת אבטיפוסים. בגישה זו מושם דגש מועט על הניסיון לקבוע את הדרישות לפני התחלת הפיתוח, ודגש רב יותר על מתן אפשרות להתגלות הדרישות תוך כדי פיתוח מספר פתרונות של אבטיפוסים. סגנון יצירת אבטיפוסים ממקד את תשומת הלב במוצר המוגמר ולא בתהליך.

יישום שיטות איכות מוכוונות-תהליכים וטכניקות מקובלות של ניהול פרויקטים, אינו בעל סיכויים רבים להצלחה בסביבה זו שיש בה הרבה איטרציות, ולכן יש צורך בגישות חדשות לניהול איכות. קבוצה חשובה ביותר זו של שיטות כוללת: יצירת אבטיפוסים, ובכלל זה **פיתוח יישומים משותף** (Joint Application Development), **פיתוח יישומים מהיר** (Rapid Application Development), **ושיטות מוכוונות אובייקטים** (object-oriented methods).

אף שעד כה לא התגלו מודלים 'סטנדרטיים' של מחזורי חיים בלתי-קוויים, ארגונים רבים פועלים בשיטות בלתי-קוויות, ומנסים להגדיר לעצמם מתודולוגיה ומחזור חיים משלהם. ספר זה טוען שהגישות 'המסורתיות' לניהול איכות לא תוכלנה לשרת בצורה נאותה את השיטות הבלתי-קוויות, משום שהתקנים עליהם מבוססות מערכות לניהול איכות, נכתבו לפי תהליכים קוויים. נושאים אלה יידונו בפרקים 7, 8, ו- 9.

פיתוח התפתחותי (אבולוציוני)

שיטות פיתוח המתפתחות תוך כדי התהליך, מכירות בעובדה שלא ניתן לנתח בצורה נאותה את הדרישות ו/או לייצגן כבר בתחילת הפרויקט. שיטות אלו מבקשות להשתמש בארכיטקטורה לבניית מערכת, שבמסגרתה יכול הפיתוח להימשך עד לשעה בה יוגדרו הדרישות וייושמו סופית. בדרך כלל, שיטות אלו כרוכות בסדרה של שלבים התפתחותיים, שכל אחד מהם מספק פונקציונליות נוספת, עד למועד בו עונים על

התפיסה של הדרישות. המוצר הראשון של מערכת כזו הוא בדרך כלל הארכיטקטורה הבסיסית, המספקת את המבנה בו ניתן לחקור את תחומי הפיתוח הצפויים. שיטות שנעשה בהן שימוש בפיתוחים מסוג זה, כוללות בדרך כלל צורה כלשהי של טכנולוגיות ליצירת אבטיפוסים. טכניקות ספציפיות יידונו בפרק 8.

פיתוח מוכוון-אובייקטים

שיטות **מוכוונות-אובייקטים** (object oriented) הן בעלות אופי התפתחותי (אבולוציוני), וגם מייצגות גישה מיוחדת בתחום הגדרת ופירוט הדרישות. כך הן יוצרות פתח למבני מערכות שונים מאלה שמחוללות השיטות ה'תבניתיות' (structured) והמקובלות יותר. אופי הגישה מוכוונת-האובייקטים מצריך 'סגנון' שונה של פיתוח, וזה מוליך אל מודלי פיתוח חדשים. בדומה לטכנולוגיית יצירת אבטיפוסים, שיטות מוכוונות-אובייקטים מחייבות מודל נפרד לניהול פרויקטים ואיכות. שיטות אלו מתוארות בפרק 9.

ניהול פרויקט תוכנה

בעיית ניהול פרויקטים לפיתוח תוכנה אינה כה מורכבת ובלתי נשלטת, כפי שיכול היה להשתמע מניסיוננו עד כה, לפחות לא לגבי פרויקטים המאמצים מחזור חיים קווי, ובתנאי שקיימת תרבות מבוססת של הנדסת תוכנה. תכנון פרויקטים משמש תמיד יסוד לניהול טוב של פרויקטים, ותוכנית פרויקט המאמצת מודל מוגדר היטב, ומאגדת את כל המשמעויות של תרגילי הפיתוח ותכנון האיכות, יש לה רוב הסיכויים להיות תוכנית מציאותית. בפרק 3 נדון בקשיים ונציע מספר פתרונות.

תוכנית פרויקט אינה יכולה להיות שלמה כבר בתחילת הפרויקט, ולכן דיסציפלינה של עדכונים סדירים היא רכיב חיוני בה. קביעה מראש של כללים ברורים וקווים מנחים למפתחים, תסייע לשמור על תרבות הפרויקט ותאפשר למנהל לזהות את הסיכונים ולנהלם. תפקיד המערכת לניהול איכותי הוא לספק אוסף נהלים, הוראות עבודה, ותקנים שיגלמו את תרבות הנדסת התוכנה. חיוני שגישת ניהול הפרויקט תבוסס על תרבות האיכות (של הנדסת התוכנה), כדי שהתוכניות תהיינה מציאותיות וכדי שחברי הצוות יוכלו לפרשן בדרך עקבית. מערכת ניהול איכותית המוגדרת היטב, מפשטת בצורה משמעותית את משימת ניהול הפרויקט.

מערכת ניהול איכותי מגלמת בתוכה את מדיניות האיכות של החברה, כמו הכרה בעובדה שזול יותר לפתור בעיות תוך כדי הפיתוח מאשר בזמן מחזור התחזוקה. בגישה זו יקטן הסיכון שמהנדסים, או מנהלי פרויקטים ינקטו בקיצורי דרך, בשל ההנחה המוטעית ש'הפריון חייב לקבל קדימות על פני האיכות'.

העדפת האיכות היא הדרך היותר משקית בחיי היומיום, אם כי רק לעיתים רחוקות מרגישים כך תוך סערת הקרב. מערכות לניהול איכות מסייעות לנו לקיים את השיקולים שהתקבלו באווירה הצוננת והרציונלית, גם בהיותנו שרויים בעיצומו של הקרב. זה אחד מ'גילויי' המפתח שהובנו לאחר התנסות במספר רב של כישלונות.

הדיסציפלינות של ניהול איכות וניהול פרויקטים אינן ניתנות להפרדה זו מזו. הן מייצגות שני רכיבים של גישה הוליסטית לנושא הניהול. אין החלטות בתחום האיכות שאינן משפיעות על פרויקטים, ואין החלטות בתחום הפרויקטים שאינן משפיעות על האיכות. מערכת לניהול האיכות נועדה ליצור סביבת ניהול בה שני פרמטרים חשובים אלה שזורים זה בזה.

גישת תקן ISO 9000-3 לניהול איכות תוכנה

הרעיונות שהוצגו בפרק זה הציגו את הצורך בגישה דיסציפלינרית להנדסת התוכנה, שמבוססת על עקרונות הנדסיים, ותואמת עם זאת גם את האופי הייחודי של פיתוח תוכנה. מפרטי האיכות והשגתם חייבים למלא תפקיד מרכזי בתוך דיסציפלינה זו.

הראינו כי להשגה עקבית של האיכות יש קשר לתהליכים בסיסיים שנעשה בהם שימוש בתִיכון ובבניית מוצרים. הרעיון של מערכת לניהול איכות התגלה כהגדרה רשמית של תהליכי התשתית, עליהם מורכבים מנגנוני איכות חיוניים, כגון אבטחת האיכות ובקרת האיכות.

מתווה זה של מערכת לניהול איכות עבור סביבת פיתוח תוכנה, מורחב בתקן ISO 9001, (או ת"י 2001 בארץ, בעתיד: ת"י ISO 9001) לכלל מודל של מערכת לניהול איכות, בתוספת קווים מנחים עבור היבטי התוכנה המגולמים בתקן ISO 9000-3 (או ת"י 2000-3 בארץ).

השיטות ה'בלתי-קוויות' מספקות גישה של פיתוח המוכוון על ידי המוצר, מה שמביא בעקבותיו קבוצה חדשה של אתגרים איכותיים. עלינו ללמוד להכיר את אופי בעיות האיכות שניצבות בפני אתגרים אלה, ואת שיטות האיכות שניתן לאמץ כדי לספק סביבת פיתוח 'בטוחה' יחסית, בה ניתן לבצע את לימוד אופי הבעיה תוך סיכון מזערי.

תחילה, לפני שנפליג במסע חקר מפורט של הבעיות שהועלו בפרק זה ושל הקווים המנחים של תקן ISO 9000-3, ננסה להרחיב את הדיבור על הערות המבוא המתייחסות למערכות לניהול האיכות. זה יהיה נושא פרק 2.

פרק 2

מערכות לניהול האיכות - לשם מה?

מבוא

מהן מערכות לניהול איכות ומדוע אנו זקוקים להן? פרק זה מתאר את הסיבות שבעטיין נחשבת האיכות לנושא בעל חשיבות ראשונית, ומציג את הדרך בה אנו מטפלים באתגר האיכות. למרות שיש ערך מסוים לכל הגישות השונות, בחרנו לתאר באופן מפורט את הנתיב של תקני 'מערכת ניהול איכות' Quality) QMS Management System), משום שלגבי ארגונים רבים זהו הצעד האקטיבי הראשון בדרך אל האיכות. סכימות כלל-ארציות, כדוגמת הסכימה UK TickIT, שמרחיבות את גישת התקן הבסיסי, מסייעות לקדם את החברות העסקיות על ידי זיהוי הזדמנויות לשיפור מערכת QMS שלהן. היעד הסופי הוא סביבת איכות כוללת, בה האיכות היא חלק בלתי נפרד מתרבות הניהול; מדידה ועריכת מבדקים (benchmarking) בהשוואה למתחרים הם נוהג של קבע; ושיפור מתמשך הוא יעד עסקי בסיסי.

אתגר האיכות

מדוע אנו זקוקים לאיכות? עשרים וחמש שנים הצלחנו להסתדר בלעדיה, אז למה כל ההתרגשות הזאת עכשיו? השימוש במילה 'להסתדר' משקף באמת את מצב העניינים הנוכחי. הולכות ומצטברות הוכחות שפיתוח תוכנה הוא מהלך יקר עד כדי ייאוש. גרוע מזה, העלות האמיתית של רכישה, בעלות ושימוש במוצרי תוכנה קריטיים, משבשת את פעולת החברות, וללא ספק חורגת ועוברת במספר רב של מקרים, אם לא בכולם, את התועלת המוחשית המושגת באמצעות התוכנה.

במרוצת הזמן הוצעו הערכות שונות למדידת העלות האמיתית של התוכנה. ההערכה לה עלינו להיות מודעים מאוד, היא זו המודדת את היחס שבין מספר האנשים שעוסקים בבניית יישומי תוכנה חדשים, לבין אלה ש'מתחזקים' יישומים קיימים. בהקשר זה, תחזוקה פירושה דאגה לכך שהתוכנה תמשיך לתפקד כמות שהיא, ללא הרחבות נוספות. היחס הוא ממש מהמם, 1:1. אנו צורכים את מחצית משאבי פיתוח

התוכנה בהתמודדות עם בעיות שנוצרו תוך כדי תהליך הפיתוח. זו הסיבה שהאיכות היא כה חשובה; אנו חייבים למצוא דרכים שתאפשרנה לנו לצמצם בהמשך הדרך את העלויות הכרוכות בבעלות על התוכנה, על ידי אספקת מוצרים טובים יותר. נושאי פרק זה הם: המשמעות של מוצרים טובים יותר, ודרכים לשפר את איכות מוצרים אלה בדרך משקית.

> ***כאשר עובד בקו הייצור עושה שגיאה, הוא משליך הצידה את הרכיב הפגום. מנהל העבודה מבקש אחר כך לדעת מדוע ערמת הפגומים היא כה גדולה. אנו עושים כל הזמן את אותו הדבר בזמן כתיבת התוכניות, אלא שאיש אינו יכול לראות את ערימת הפגומים.***
>
> **יאן גוסטפסון (SQM, 1993)**

מהי הדרך אל האיכות

כדי להשיב על שאלה זו עלינו להתמודד עם מספר משתנים. ראשית, יש דעות מנוגדות בדבר **מהות האיכות והדרכים להשגתה**; את אלו יש לבחון בתשומת לב. שנית, עלינו להבהיר לעצמנו **למה אנו מתכוונים כאשר אנו מדברים על איכות תוכנה**. שלישית, עלינו לשאול את עצמנו **מה הדבר שמייחד את התוכנה כל כך**, מדוע יש לנהוג בה בדרך שונה מזו שבה אנו נוהגים בכל מוצר אחר?

לאחר שנשאל את עצמנו את השאלות האלו, עלינו לקבל החלטה. עלינו להחליט מה לעשות כדי לפתור את הבעיה של תוכנה שאיכותה ירודה, ואנו חייבים להיות מסוגלים להוציא אל הפועל את דרך הפעולה בה בחרנו. איננו מחפשים פתרון אידיאלי, או פתרון שייתן לנו את מירב היתרונות, משום שדבר זה יחייב זמני עיצוב ויישום ממושכים. הצורך הוא דחוף, ולכן מטרתנו מוגבלת יותר והיא, בחירת דרך פעולה שאפשר להתחיל בה מיד ואשר תביא לשיפור מהיר של המצב.

מאוחר יותר נדון באחדות מאפשרויות הטווח הארוך יותר. עלינו לעשות זאת כדי לוודא שהאסטרטגיה שלנו סבירה ועקבית, ואמנם מטפלת ביעד החשוב ביותר - היכולת לשיפור מתמשך. עם זאת, בשלב זה תהיה גישתנו פרגמטית בלבד.

מהי איכות תוכנה

נאמנים לגישתנו הפרגמטית, לא נחפש הגדרה מחמירה. ננסה רק ללכוד אחדות מהסוגיות המעשיות הנוגעות לנושא האיכות.

מה אנו מחפשים

כיצד אנו יכולים להבחין באיכות שבמוצר תוכנה?

1. המוצר מבצע את מה שמצפים ממנו שיבצע.
2. המוצר אינו מבצע את מה שלא מצפים ממנו.
3. המוצר מתנהג בדרך עקבית.
4. המוצר מתנהג בצורה אמינה.
5. המשתמשים במוצר יכולים להשתמש בו בדרך אפקטיבית.
6. המוצר ניתן לשינוי בקלות יחסית, להחלפה או להרחבת הפונקציות שלו.

רשימה זו של סוגיות רחוקה מלשמש כהגדרה. עם זאת, אין ספק שאילו מספר גדול יותר של מוצרי תוכנה היה עונה על הדרישות הלגיטימיות שפורטו לעיל, לא היינו צריכים להיות מוטרדים כל כך בנושא האיכות.

כיצד נוכל להתחיל לענות על הסוגיות האלו? ראשית, עלינו לנסות להבין את מהותן. נשים לב שניתן לאגד את הסוגיות האלו בארבע קטגוריות:

- סעיפים 1 ו- 2 עוסקים במתן מענה לציפיות;
- סעיפים 3 ו- 4 עוסקים בהתנהגות הכוללת - הם משקפים את איכות התיכון;
- סעיף 5 עוסק בתחושות המשתמש;
- סעיף 6 עוסק בפוטנציאל התיכון לטווח ארוך.

מתן מענה לציפיות והשגת התחושות הנכונות של המשתמש, כרוכים בתקשורת יעילה ומתמשכת עם הלקוח שלנו, כדי לוודא שהגענו להבנה נכונה ושלמה של ציפיות ותחושות הלקוח. זה מחייב שיעמוד לרשותנו תהליך כלשהו להמרת הבנה זו לאיזה שהוא מוצר תוכנה מוגמר. אם נרצה להגיע לכלל איכות בצורה מתמשכת ועקבית, נזדקק לתהליך שניתן יהיה לחזור עליו.

איכות התיכון ופוטנציאל העיצוב הם בעלי אופי שונה - הם מותנים בכישורים ובמקצועיות המעצבים שלנו, ותלויים פחות בתהליך עצמו. את התיכון הטוב אי אפשר לבטא בצורת תיאור נהלי פשוט, אם כי גם כאן ניתן להסתייע בתהליך תיכון עקבי.

אם כן, תהליכים שעליהם ניתן לחזור, אינם יכולים לפתור בשלמות את בעיותינו, עם זאת, אין ספק שיש בכוחם למלא תפקיד נכבד. תפקיד זה חשוב, משום שהאלמנטים הנהליים של בעיותינו קשורים ל**עקביות, נכונות ושלמות** (consistency, correctness and completeness). תהליכים שניתן לחזור עליהם, אין בהם כדי להבטיח את העיצוב הטוב, או להעניק נופך יצירתי. עם זאת, הבעיות הקשורות במוצרי תוכנה מתמקדות, בדרך כלל, בנושאי האמינות הבסיסית וביכולת התחזוקה של המוצר. מה שדרוש לנו הוא קבוצה של נהלי איכות אשר יסייעו במאבק להשגת העקביות.

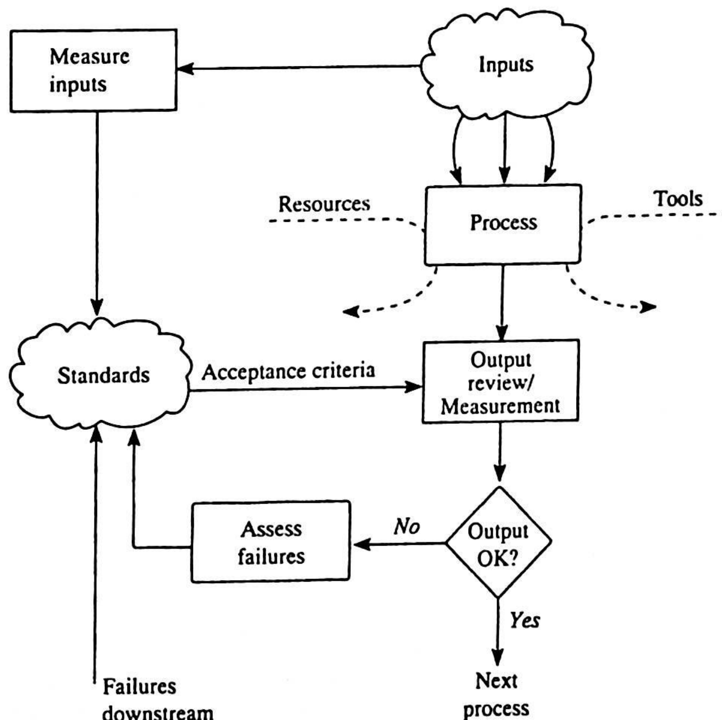
תהליכי איכות

למה אנו מתכוונים במילים **תהליך איכות** (Quality process)? כוונתנו לתהליך המאגד מידה מספקת של **פעולות אימות** (verification), כדי לוודא שהושלם בצורה נכונה, והוא מכיל מנגנון כלשהו לבדיקת **תקפות** (validation), כדי לוודא שהוא מספק מוצר קביל. גישה זו אינה מחוסנת לחלוטין מפני פגמים, אך מהווה צעד בכיוון הנכון.

תהליך איכות צריך להכיל את כל האלמנטים המוצגים בתרשים 2.1, או חלקם. פלט התהליך מושווה מול איזה שהוא מדד של מבחן קבלה. כל פלט שאינו עומד במבחן מדדים זה ייפסל, או יועבר לעיבוד חוזר. המדדים למבחן הקבלה יכולים להתבסס על מידע השאוב משלושה מקורות:

- הקלט המוזן לתהליך, כדי לוודא שהבאנו בחשבון תנודות באיכות הקלט;
- תקנים, המייצגים את רמת האיכות ותואמים את המדיניות הכוללת שלנו;
- מידע המתקבל מתהליכים מאוחרים יותר, כדי שנוכל להביא בחשבון תקלות כלשהן שיהסתננו דרך הרשת'.

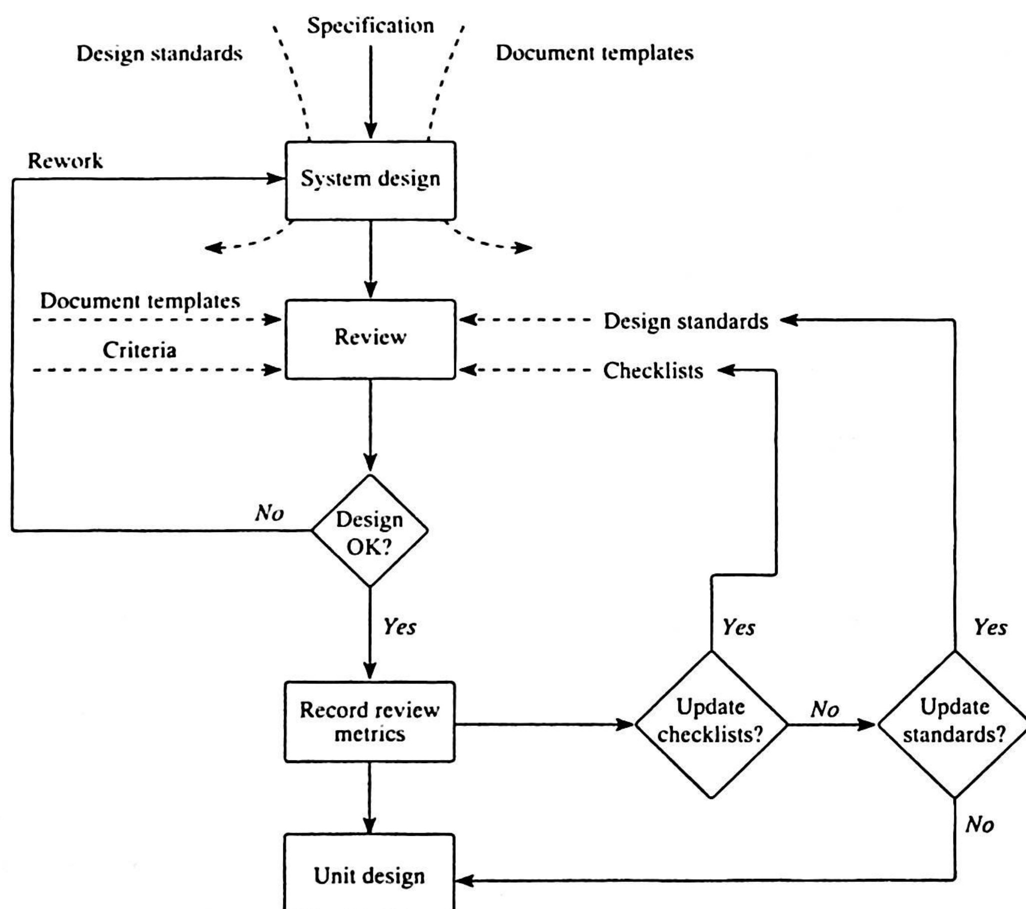
לתהליך המוגדר בצורה זו, יש את היכולת להשיג פלט עקבי, גם לנוכח תנודות באיכות הקלט.



***תרשים 2.1** תהליך איכות*

רעיון זה יכול לשמש כמודל לתהליך עיצוב האיכות, כפי שמוצג בתרשים 2.2, בו השתמשנו **בסקר התיכון** (design review) כדי לבחון את איכות הפלט המתקבל מתהליך התיכון. במקרה זה משתמש הסקר בתקנים ובתבניות שהשתמש המעצב, כדי לוודא שמסמך התיכון המוגמר תואם להם. הסקר משתמש גם בקריטריונים של מבחני הקבלה כדי לוודא שפלט התיכון **נכון**, והוא משתמש **ברשימות תיוג** (checklists) כדי להנחות את הבודקים בחיפושיהם אחר חולשות פוטנציאליות. רשימות התיוג הן מנגנון רב עוצמה לרישום מידע, המתייחס לתקלות העבר באמצעות שאלות המנחות את הבודקים לחפש אחר סימפטומים אופייניים של בעיות שכיחות. רישום יסודי של התוצאות יאפשר לנו להשתמש בממצאים לבדיקה מחדש של התקנים ורשימות התיוג שבידינו, כדי לשמור על העדכניות והרלוונטיות לסוגי הבעיות בהן אנו נתקלים.

מתוך התהליכים שתוארו, משתמעים התקנים העוסקים בתוכן **ובמתווה** (layout) מסמכי מפתח. תהליכי איכות צריכים להגדיר את מה שחייב להיעשות ואת הדרך בה נוכל לדעת אם התהליך הושלם בצורה נכונה.



***תרשים 2.2** תהליך עיצוב איכות*

מערכות איכות

מערכת איכות (Quality system) היא מערכת בה כל התהליכים הם תהליכי איכות, ואף למעלה מזה. עד עתה תיארנו בתהליך האיכות שלנו רק כיצד צריכה העבודה להתבצע ולהיבדק, אך יש גם סוגיות אחרות בהן עלינו לעסוק, למשל, מה יקרה כאשר משתתפים בביצוע העבודה יותר מאשר עובד אחד? על מי תוטל האחריות לכך שהדברים ייעשו בדרך הנכונה? ברור שכדי להתגבר על חולשה זו עלינו להקצות תחומי אחריות בתוך התהליכים עצמם.

ומה יקרה כאשר התהליך כרוך בתקשורת בין מחלקות שונות, או בינינו לבין הלקוח? בדרך כלל, אלה תחומים שחשופים לתקלות, ולכן עלינו להקדיש תשומת לב מיוחדת להגדרת ממשקים אלה. אנו חייבים לוודא שתהיה הגדרת אחריות לניהול התקשורת על פני ממשקים, ועלינו גם לוודא שהגדרנו את התקשורת הדרושה, כדי שהתהליך יוכל להסתיים בהצלחה.

הרחבנו את הדיבור על רעיון תהליכי איכות כדי שיכללו נהלים שמטרתם לקיים בקרה על הממשקים והקצאת תחומי אחריות לכל התהליכים ונהלי הממשקים. עתה יש ברשותנו דבר שנוכל לתאר במידה מסוימת של סבירות כמערכת, משום שהוא מכסה את כל התהליכים והממשקים העיקריים.

תחום שטרם כיסינו הוא רכיב הניהול שימיר מערכת איכות זו ממעמד של מנגנון לשיפורים טכניים, למעמד של מערכת אמיתית החובקת את כל מה שאנו מבצעים, ומעודדת יצירת תרבות אפקטיבית ועקבית של איכות. מחויבות ההנהלה היא הדבק המאחד את מערכת האיכות לכלל שלמות אחת. ההנהלה מגדירה את מדיניות האיכות ואת היעדים, מקצה תפקידים ותחומי אחריות, ומספקת משאבים כדי לאפשר לתהליכי האיכות לתפקד ביעילות. ההנהלה מחליטה אם לתמוך (או לא לתמוך) באיכות, כאשר לחצים מסחריים מעוררים את הפיתוי להשתמש בקיצורי דרך. מערכת הפועלת בדרך זו ומצליחה לעמוד בפני לחצים היא באמת מערכת לניהול איכות.

מערכות לניהול איכות

מערכת לניהול איכות היא אם כן מערכת המכילה את כל הרכיבים העיקריים:

- מחויבות ההנהלה;
- תהליכי איכות מוגדרים, כולל תקנים למוצר המוגמר;
- תחומי אחריות מוגדרים;
- הגדרת הממשקים העיקריים;
- מנגנוני אימות לבדיקת התהליכים;
- מנגנונים לבדיקת תקפות לצורך בדיקת המוצרים.

השאלה הבאה עוסקת בדרך בה מיישמים את המערכת. לפנינו שלוש אפשרויות:

- לבחור עובדים מומחים במיוחד, לומר להם מה נדרש מהם, ולהניח להם למצוא את הדרך להשגת התוצאות הרצויות.
- להדריך את העובדים בשימוש בשיטות ובתהליכים שאנו רוצים, ולהניח להם למצוא את הדרך להשגת הרמה הרצויה של שיתוף הפעולה.
- לתעד את כל נהלי ותהליכי האיכות, הממשקים ותחומי האחריות, ולדרוש מהעובדים לפעול לפי המערכת המתועדת.

שלוש גישות אלו כבר נוסו בעבר, ואפשר למצוא בענף התכנות דוגמאות של כל אחת מהן. אין כל עדות תקפה שתאמר איזו מהגישות עדיפה משמעותית על האחרות. עם זאת, האפשרות של יצירת מערכת איכות מתועדת, גובשה כבר בתקן הבינלאומי - ISO 9001 (או ת"י 2001 בארץ). נוסף לכך שתקן זה מספק מודל למערכת ניהול איכות, הוא גם מזהה נהלים ספציפיים שיש לתעד בכל מקרה כמינימום. רמת פירוט ההגדרה והיכולת הנלווית אליה מאפשרות להעריך את מידת ההיצמדות למפרטי מערכת האיכות, ולכן גרמו לארגונים רבים לבחור בתקן זה כאפשרות המועדפת.

היתרון הראשי של מערכת איכות מתועדת פשוט למדי: המערכת המתועדת יכולה לשמש כנקודת מוצא מוחשית. בכל מקרה שמתעוררות בעיות איכות, ניתן לנתח את המערכת המתועדת ולשנותה, כדי לתקן את הבעיה ולהימנע מהישנותה בעתיד. לאחר שהמערכת תתבסס היטב, ומוצרי איכות יהיו לדבר שבשיגרה, אפשר יהיה להשתמש במערכת המתועדת כבקרש קפיצה לשיפורים עתידיים. הצגת תיאור שלם של המערכת הנהוג לפי גישה זו מקנה לה יתרון מכריע על פני כל הגישות האחרות.

מאידך, מערכות איכות מתועדות יכולות לעודד הסתגלות עיוורת לנהלים בלתי יעילים וביורוקרטיים. אין ספק שבכל מערכת פורמלית מובנית אינרציה מסוימת, ותקנים עלולים להניע את האנשים להסתפק במינימום מסוים, במקום לשאוף מעלה, אל הטוב ביותר שניתן להשגה. עם זאת, חולשות אלו אינן בלתי נמנעות, וארגון בעל הרגשת מחויבות יכול להימנע ממהמורות אלו.

> ***הימנע מלבנות אימפריות של איכות. אלה משמשות רק כתירוץ לאי השגת האיכות.***
> ***קמרון לאו (SQM 1993)***

סדרת התקנים ISO 9000

ISO 9000 (בארץ, ת"י 2000) היא משפחה של תקנים וקווים מנחים שנועדה לתת תמונה כוללת של ניהול איכות, להציב תקנים בהקשרם הנכון, ולספק הדרכה לגבי חלק מהפירוט הנדרש ליישום מערכות אפקטיביות לניהול איכות. התקנים ISO 9001, ISO 9002, ו-ISO 9003 (בארץ, ת"י 9001, ת"י 9002 ות"י 9003 בהתאמה), מספקים תבניות שעל פיהן אפשר לשפוט את המערכות לניהול איכות. מתוך תקנים אלה, תקן ISO 9001 הוא הכוללני ביותר וישים בכל מקרה שמערכת לניהול איכות עוסקת בתִיכון ובפיתוח.

דרישות תקן ISO 9001

הדרישות המפורטות של תקן ISO 9001 מתוארות במסמך 1994, BS EN ISO 9001 (בארץ ת״י ISO 9001). אנו כוללים כאן את סיכום הדרישות כדי לאפשר השוואה עם המודל הכללי של מערכת ניהול איכות שהוגדרה על ידינו.

> ***מהנדסים אזרחיים חייבים לפעול לפי תקנים, אחרת יתמוטטו הגשרים שלהם. בתחום התוכנה צריכים עוד ליפול הרבה גשרים עד שנלמד את הלקח.***
>
> ***לס האטון***

דרישות תקן ISO 9001 מאורגנות בצורה שנועדה לשקף את הייצור, מכיון שהתקנים נכתבו מלכתחילה עבור המגזר התעשייתי. כדי להבין את מה שהתקן מנסה להשיג, מן הראוי לשקול שלושה תחומים של סוגיות עיקריות בהן הוא עוסק.

- אחריות ההנהלה;
- ארגון וניהול העבודה עצמה;
- פעילויות התמיכה המאפשרות את נהגי העבודה ומספקות שיפורים מתמידים.

התקן מכיל 20 סעיפים אשר ממוספרים במספרי סעיפים מ- 4.1 ועד 4.20. הדיון שבהמשך מתייחס אל מספרי הסעיפים האלה.

ISO 9001 - 20 הסעיפים

4.1 אחריות הנהלה (management responsibility).

4.2 מערכת האיכות (quality system).

4.3 סקר החוזה (contract review).

4.4 בקרת התיכון (design control).

4.5 בקרת התיעוד (document and data control).

4.6 רכש (purchasing).

4.7 בקרת מוצר שמסופק על ידי הלקוח (control of customer-supplied product).

4.8 זיהוי המוצר ועקיבותו (product identification and traceability).

4.9 בקרת תהליך (process control).

4.10 בחינה ובדיקה (inspection and testing).

4.11 בקרת ציוד הבחינה, המדידה והבדיקה (control of inspection, measuring and test equipment).

4.12 מצב הבחינה והבדיקה (inspection and test status).

4.13 בקרת מוצר לא-מתאים (control of nonconforming product).

4.14 פעולה מתקנת ומונעת (corrective and preventive action).

4.15 שינוע, אחסון, אריזה, שימור והספקה (handling, storage, packaging, preservation and delivery).

4.16 בקרת רשומות איכות (control of quality records).

4.17 מבדקי איכות פנימיים (internal quality audits).

4.18 הדרכה והסמכה (training).

4.19 מתן שירות (servicing).

4.20 טכניקות סטטיסטיות (statistical techniques).

אחריות הנהלה

אחריות ההנהלה היא הנושא הראשון בו עוסק התקן. סעיף 4.1, אחריות ההנהלה, כולל דרישות שמחייבות להציג בבהירות את מדיניות האיכות של הארגון (4.1.1), להגדיר את הארגון שבמסגרתו אמורים להשיג את האיכות (4.1.2), ולסקור דרך קבע את ביצועי מערכת ניהול איכות (4.1.3).

מדיניות האיכות מוגדרת ב'**מדריך האיכות**' (Quality Manual), שמפרט את הרמה הגבוהה ביותר של תיאור מערכת ניהול איכות. כל העובדים אמורים להיות בעלי אחריות מוגדרת היטב, ובעלי הסמכות הנדרשת להפעלת האחריות. נציג ההנהלה חייב להיות בעל אחריות כוללת לנושא האיכות.

התקן מחייב קיום **סקרי הנהלה** שגרתיים, כדי לוודא את המשך קיום '**מידת ההתאמה והאפקטיביות**' של המערכת לניהול איכות.

בתקן כלולים שלושה סעיפים חשובים נוספים, בהם מוצגים היבטי אחריות ההנהלה:

- סעיף 4.2 מציין את הדרישה לתעד את מערכת ניהול איכות.
- סעיף 4.14 דורש נקיטת פעולה מתאימה לתיקון בעיות במוצרים ובתהליכים.
- סעיף 4.17 דורש לקיים מבדקים מתוכננים תקופתיים של מערכת ניהול האיכות.

היבטים אלה של התקן יתוארו בפירוט רב יותר בפרק 5.

תחום העבודה

תהליכי האיכות של חברה עוקבים אחר זרימת העבודה בעסק.

סקר החוזים (4.3) מכסה את כל הפעילויות המשפיעות על הקשרים החוזיים עם לקוחות, כולל ניתוח הדרישות וכל עבודה הנדרשת, כדי לפתור בעיות הקשורות בעמידה בדרישות אלו.

בקרת התִיכון (4.4) ובקרת תהליכים (4.9) עוסקות בכל ההיבטים של תִיכון וייצור המוצרים. במשך שלבי התִיכון והייצור יש צורך לשקול בחינות ובדיקות (4.10) של מוצרי ביניים ומוצרים סופיים, שעשויים לחייב את השימוש בציוד בדיקה מכויל (4.11). בעיות כלשהן המתגלות במשך שלבי התִיכון והייצור, מחייבות תשומת לב מיוחדת, כדי לוודא שרק מוצרים קבילים יגיעו אל הלקוח, ומוצרים שאינם תואמים את התקן יחויבו בבקרה מיוחדת (4.13).

כל בקרה הנדרשת במשך שלבים אלה, מתאפשרת על ידי זיהוי דקדקני של פריטי הייצור (4.8) והשלבים שעובר כל אחד מהם (4.12).

שינוע, אחסון, אריזה ומשלוח (4.15) קובע את התקנים אשר נועדו לוודא שהמוצרים יגיעו אל הלקוח במצב הנכון, ומתן השירות השוטף (4.19) מגדיר קשרים נמשכים עם הלקוח שעשויים להתקיים גם לאחר שהמוצר סופק.

פעילויות תומכות

כדי שאפשר יהיה לוודא את האפקטיביות של מערכת ניהול איכות, יש צורך במספר פעילויות תומכות.

בקרת תיעוד (4.5) מחייבת שהתיעוד הרלוונטי יהיה נתון לבקרה כדי לוודא שהעותקים הנכונים של המסמכים הרלוונטיים יהיו זמינים בכל מקום בו הם נדרשים.

רשומות האיכות (4.16) עוקבות אחר פעילויות האיכות ומספקות עדות ליישום האפקטיבי.

הרכש (4.6) מזהה את הצורך בבחירת ספקים ומוודא שפריטים שנרכשו נבדקים עם קבלתם ומוגדרים בדייקנות. מוצר המסופק על ידי הלקוח (4.7) מרחיב את הטיפול בחומרים עבור חומרים השייכים ללקוחות ומוחזקים עבורם בחצרות הספק.

הדרכה (4.18) קובעת את הדרישות לזיהוי הדרכה חיונית ומוודאת שצורכי ההדרכה ומתן ההדרכה יימצאו דרך קבע תחת בקרה.

אחרון אך לא פחות חשוב, טכניקות סטטיסטיות (4.20) מצביעות על הצורך למדוד את מאפייני המוצר והתהליך, כדי לאפשר שיפורים מתמידים.

מה מיוחד כל כך בתוכנה

תקן ISO 9000 חל באופן **שווה** על כל ענפי התעשיה. זהו תקן כללי, אשר מכוון אל כל המרכיבים של מערכות איכות, לאו דווקא לתהליכים המונחים בתשתיתם. לרוב ענפי התעשייה הפרשנות לתקן זה מצומצמת למדי, אך האופי של תהליך פיתוח התוכנה עושה את פרשנות התקן בנושא זה לקשה יותר. קיימות שלוש סיבות לכך:

1. התהליך מפיק פונקציונליות או התנהגות, וכמעט שום דבר אחר מלבד זאת. תהליך הייצור עצמו טריוויאלי; תהליך העיצוב הוא לב לבו של פיתוח התוכנה.
2. צורת תהליך הפיתוח - מחזור החיים - היא בעלת חשיבות מיוחדת, וחובה להתאימה בקפדנות לפיתוח בו עוסקים. דגש זה על מחזור החיים של הפיתוח חייב להשתקף במסמך ההנחיות.
3. פעולות מסוימות, כגון ניהול התצורה, ממלאות תפקיד תומך וחשוב, וצריך להדגישן ולהתייחס אליהן במונחים המוכרים למפתחי תוכנה.

מסיבות אלו, גובשה קבוצה של קווים מנחים שנועדו להשלים את תקן ISO 9001.

הקווים המנחים ISO 9000-3

תקן ISO 9000-3 (בארץ, ת"י 2000-3) - בנוי בצורה שונה מזו של ISO 9001. הדבר נעשה בחלקו כדי לאפשר את שינוי הסדר, תוך התאמה לרעיון של מחזור החיים של הפיתוח, ובחלקו בשל הוספת נושאים חדשים, שעבורם אין סעיף מקביל בתקן ISO 9001. תרשים 2.3 מציג את הקשר בין המבנים ואת המקומות בהם ISO 9000-3 מציג חומר חדש.

המטרות העיקריות של ISO 9000-3 הן: קידום היבט חדש של ניהול איכות המונחה על ידי מחזור החיים והדגשת תחומים מסוימים בהם יש צורך בדגש מיוחד בהקשר של פיתוח תוכנה.

ISO 9000-3 - פירוט הסעיפים

4 מערכת איכות - מסגרת:

4.1 אחריות הנהלה (management responsibility).

4.2 מערכת איכות (quality system).

4.3 מבדקים פנימיים של מערכת האיכות (internal quality system audits).

4.4 פעולה מתקנת (corrective action).

5 מערכת איכות - פעילויות במחזור החיים:

5.1 כללי (general).

5.2 סקר החוזה (contract review).

5.3 מפרט דרישות הלקוח (purchaser's requirements specification).

5.4 תכנון הפיתוח (development planning).

5.5 תכנון האיכות (quality planning).

5.6 תִיכּוּן ויישום (design and implementation).

5.7 בדיקה והוכחת תקֵפות (testing and validation).

5.8 קבלה (acceptance).

5.9 שכפול, מסירה והתקנה (replication, delivery and installation).

5.10 תחזוקה (maintenance).

6 מערכות איכות - פעילויות תומכות:

6.1 ניהול תצורה (configuration management).

6.2 בקרת מסמכים (documend control).

6.3 רשומות איכות (quality records).

6.4 מדידה (measurement).

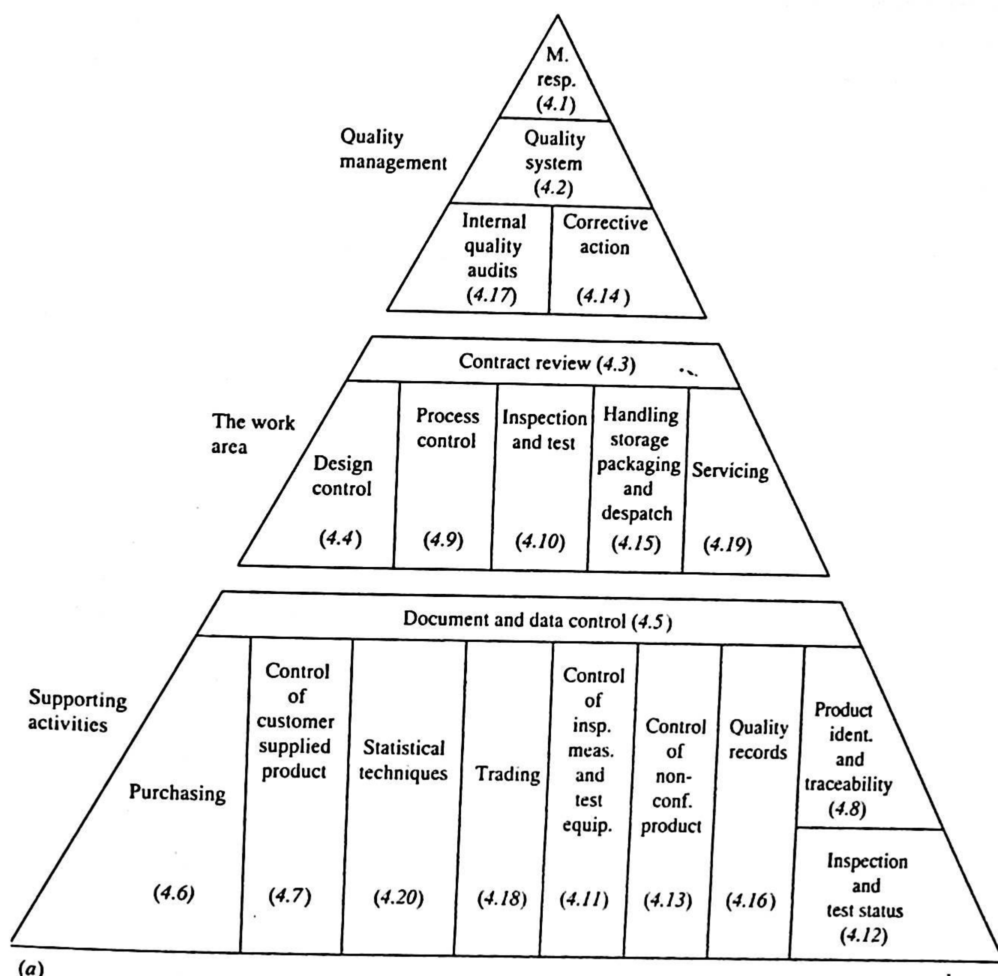
6.5 כללים, נהגים ומוסכמות (rules, prectices and conventions).

6.6 כלים וטכניקות (tool and techniques).

6.7 רכש (purchasing).

6.8 מוצר תוכנה משולב (included software product).

6.9 הדרכה (traning).

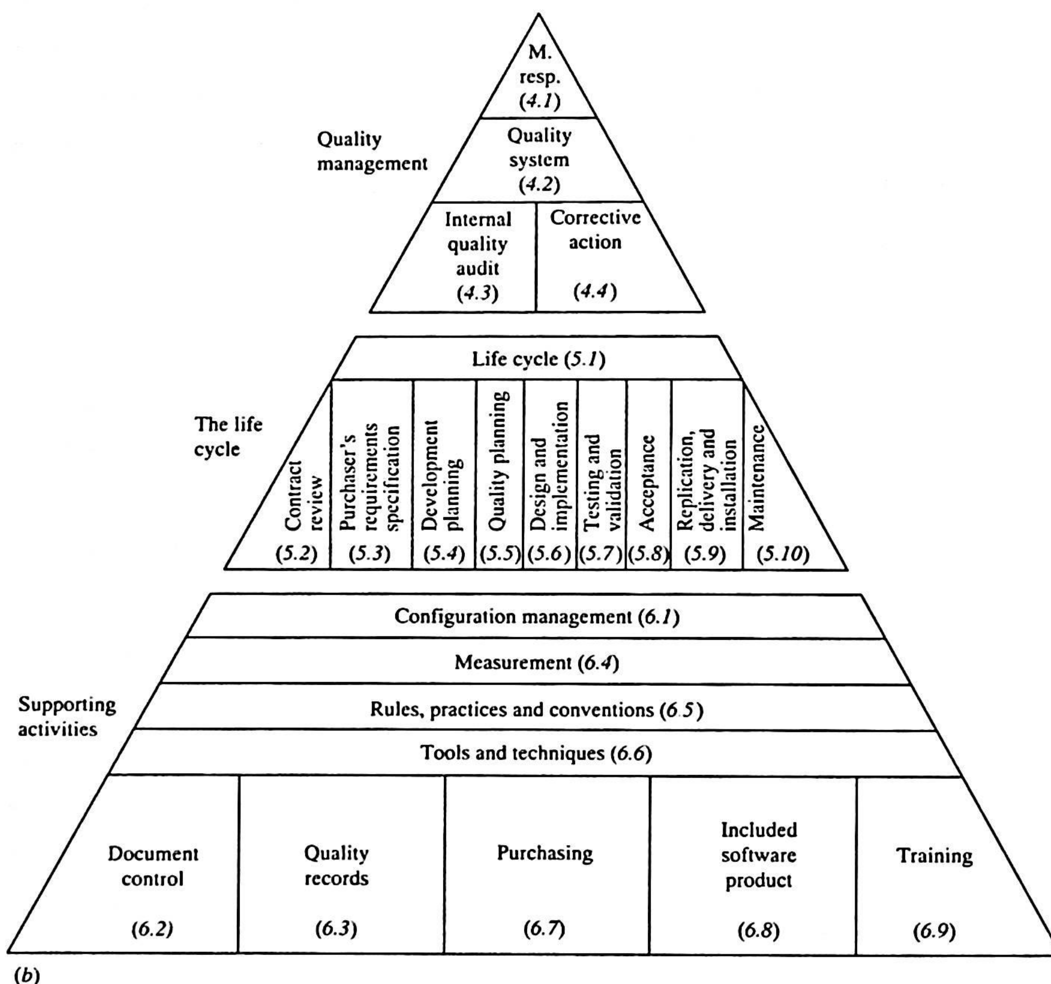


תרשים 2.3 א) מבנה תקן ISO 9001

ISO 9000-3: מערכת איכות - מסגרת

ההיבטים של אחריות ההנהלה הכלולים בתקן ISO 9000-3 דומים לאלה שבתקן ISO 9001. אותה קבוצת רעיונות הקשורים זה לזה נכללת תחת הכותרת **'מערכת איכות - מסגרת'**. בתוספת ציון של מספר נקודות.

התוספות החשובות ביותר מכוונות להשגת שיתוף פעולה גדול יותר בין הספקים לבין הלקוחות, על ידי שימת דגש על אחריות ההנהלה של הלקוח (4.1.2) ועל ידי המרצה לביצוע סקר חוזר בשלבים קריטיים של פיתוח פרויקטים (4.1.5). תוכניות איכות אמורות להיות חלק ממערכת איכות מתועדת (4.2.3).



תרשים 2.3 ב) *מבנה תקן ISO 9000-3*

ISO 9000-3: מערכת איכות - פעילויות במחזור החיים

המטרה כאן היא להפנות את תשומת הלב לאופי דמוי מחזור-חיים של תִיכון התוכנה והפיתוח בכללותם, ולהדגיש את אזורי הסיכון המחייבים תשומת לב מיוחדת בפיתוח התוכנה. תחת הכותרת '**מערכת איכות - פעילויות במחזור-החיים**', מופיע ארגון מחדש רדיקלי של סעיפי תקן ISO 9001 יחד עם תוספות משמעותיות.

סעיף סקר החוזה (5.2) כולל הנחיות בדבר מה שיש לבדוק (5.2.1) והנחיות ספציפיות בדבר **סעיפי איכות בחוזה** (5.2.2). סעיף חדש בדבר **מפרט דרישות הלקוח** (5.3) כולל יעוץ נוסף בדבר **שיתוף פעולה הדדי** (5.3.2).

תכנון הפיתוח (5.4) כולל הנחיות בדבר ההיבטים הטכניים וניהול פרויקטים. מוסיף קטע שעוסק ב**שיטות כלים לפיתוח** (5.4.2.3) ומספק הנחיות בדבר סוגי הקלט והפלט בשלבי הפיתוח. **תכנון האיכות** זוכה לסעיף נפרד משלו (5.5), וכולל הנחיות בדבר **תוכנית האיכות** (5.5.2).

תִיכון ויישום (5.6) מתייחס ישירות לפיתוח תוכנה, וקורא באופן משמעותי לעריכת סקירות של התִיכון המתועד כחלק בלתי נפרד מתהליך הפיתוח.

בדיקה והוכחת תקפות (5.7) מדגיש את **תכנון הבדיקה** (5.7.2) ומתייחס למינוח ולנהגים המקובלים בתעשיית התוכנה, **מבחני הקבלה** זוכים לסעיף נפרד (5.8).

סעיף בנושא **השכפול, במסירה וההתקנה** (5.9) עוסק בסוגיות המיוחדות של תעשיית התוכנה. בדיקות שנועדו לוודא את נכונות העותקים וכי הובאו בחשבון סוגיות של זכויות יוצרים והרשאות, מהוות סוגיות ראשיות שיש לטפל בהן לפני שאפשר יהיה להמשיך בהספקה ובהתקנה.

בסוף חלק זה של הקווים המנחים יש סעיף ראשי חדש על **תחזוקה** (5.10), אשר דן בשלב חשוב זה שבמחזור החיים של התוכנה, נושא שאינו מטופל כראוי ב- ISO 9001.

ISO 9000-3: מערכת איכות - פעילויות תומכות

תחת הכותרת '**מערכת איכות - פעילויות תומכות**', מזהה התקן ISO 9000-3 אחדים מתחומי המפתח שאינם קשורים לשלבים מוגדרים במחזור החיים, אך מהווים נושא לתשומת ליבו הכללית של מפתח התוכנה.

אחדים מתחומים אלה אינם אלא הד לתקן ISO 9001 :

- בקרת מסמכים (6.2) ;
- רשומות איכות (6.3) ;
- רכש (6.7) ;
- הדרכה (6.9).

תחומים אחרים הם חדשים, ומייצגים סוגיות ייחודיות חשובות :

- כללים, נהגים ומוסכמות (6.5) ;
- כלים וטכניקות (6.6) ;
- מוצר תוכנה משולב (6.8).

שני תחומים מייצגים תוספות עיקריות לגישת התקן :

1. ניהול תצורה (6.1) עוסק בתכנון, זיהוי ויכולת המעקב, בקרת שינויים ודיווח המצב (status) בפירוט מסוים.
2. מדידה (6.4) עוסקת גם במדידת המוצר וגם במדידת התהליך, ומרחיקה לכת מעבר לתקן ISO 9001, משום שהיא מנסה לקדם את תרבות השיפורים המתמידים.

מערכת לניהול איכות למפתחי תוכנה

התקנים ISO 9001 ו- ISO 9000-3 מספקים מפרט למערכת ניהול איכות שמנסה לענות על הצרכים המיוחדים של תעשיית התוכנה. ISO 9001 מכיל גישה בדוקה ומנוסה לניהול איכות. ISO 9000-3 מנסה להדגים את מידת הרלוונטיות של סעיפי ISO 9001 עבור מפתחי תוכנה, ומציע קווים מנחים ליישומם.

פרקים 3, 4 ו- 5 בוחנים את הקווים המנחים המוצעים על ידי ISO 9000-3 ומספקים מספר עצות שימושיות בדבר הדרכים ליישום תקן זה בסביבת פיתוח תוכנה שגרתית.

יותר ויותר פונים המפתחים אל שיטות פיתוח פחות שגרתיות, שתאפשרנה להם להשיג סבב מהיר יותר ופריון גבוה יותר. טכניקות כגון שימוש באבטיפוס ובפיתוח מוכוון-עצמים עלולות לאיים על מערכת ניהול איכות המתבססת על תהליכים שגרתיים; או מאידך, מערכת ניהול איכות תגרום לדיכוי האלמנטים החדשניים של הגישות המהפכניות לנושא הפיתוח. האם יש מקום למציאת פשרה בין השתיים?

פרקים 7, 8, 9 ו- 10 חוקרים סוגיות אלו.

איכות תוכנה - עובדות החיים

תוך כדי עיון בפרטים, מן הראוי שלא נתעלם ממספר עקרונות יסוד המאגדים את השאיפה שלנו להנדסת תוכנה יעילה, ואת הדאגה לאיכות.

מערכת איכות לפיתוח תוכנה שהיא גם מעשית וגם מציאותית חייבת להתייחס לעקרונות בסיסיים כדוגמת העקרונות של איכות התוכנה שהצגנו, ועקרונות הנדסת התוכנה שהוצגו בפרק 1. בפרקים הבאים נשתדל לזהות ולתאר את האלמנטים הדרושים כדי שנוכל להיצמד בפועל לעקרונות שקולים, יחד עם תאימות לתקן של מערכת ניהול איכות.

עקרונות איכות תוכנה

1. נוהלי איכות אינם יכולים להשיג דבר, אלא אם כן תיישמם בפועל.
2. השינויים בלתי נמנעים, בעיקר בתחום הדרישות. הבא אותם בחשבון בעת התכנון!
3. פיתוח תוכנה הוא תהליך איטרטיבי, תכנן תמיד לקראת חזרה על פעולות. עדיף ומהיר יותר להכין טיוטה מהירה שממנה יסולקו השגיאות בהמשך, מאשר לנסות להגיע לתוצאות נכונות כבר בסיבוב הראשון.
4. ייתכן שהפקת תוכנה באיכות טובה נמשכת זמן רב יותר, אך תוכנה זו אכן תפעל כאשר תגמור לבנותה, ותוכל גם לשנותה כאשר תצטרך לעשות זאת.
5. קל ומהיר יותר להפיק מסמכים או קוד באמצעות תבנית מאשר בלעדיה, ואין בכך כדי לפגום ביצירתיות שלך.

6. סקרים עולים זמן וכסף, אך הם יכולים לגלות שגיאות יקרות מאוד, כאשר תיקון השגיאות האלו הוא עדיין זול יחסית.
7. כלים וטכניקות עולים כסף רב. ללא הדרכה מתאימה הם רק צעצועים, ואפילו צעצועים יקרים מאוד.
8. אם אין ביכולתך לתכנן ניסוי עבור מפרט מסוים, יהיה זה נבון יותר אם לא תעצב אותו כלל.
9. מפרטי בדיקות הנכתבים ממש לפני ביצועם, יבדקו את התוכנה שכתבת, במקום שיבדקו את התוכנה שהיית אמור לכתוב.
10. הבדיקות יקרות וגוזלות זמן. כשלונות במתן השירות יקרים יותר, גוזלים זמן רב יותר, וגם אינם אהודים על הלקוחות.
11. הלקוחות אומרים תמיד שהם רוצים את התוכנה כבר מחר. מה שהם אומרים בפועל הוא, שהם רוצים את התוכנה בהקדם וכי הם רוצים שהיא גם תתפקד.
12. פיתוח תוכנה ללא תוכניות עבודה, הוא תמיד יותר משעשע ויותר מרגש, כל עוד אינך צריך לחשוש מפני הבעיות בפניהן תעמוד כאשר תמסור (אי פעם) את המוצר.

חלק 2

יישום ניהול איכות תוכנה באמצעות ISO 9000-3

חלק 2 של הספר עוסק באיכות התוכנה בסביבה 'קווית'. אף שיש לצפות לכך שבשלב כלשהו יהפוך פיתוח התוכנה לבעל אופי איטרטיבי, כלומר תהליך הכולל בתוכו חזרות בספר זה אנו מגדירים את הפיתוח ה'קווי' בתור מחזור חיים של פיתוח שהוא ברובו המכריע בעל אופי רציף. מחזור החיים הקרוי 'מפל המים' הוא דוגמה לכך. אחר כך נעמת גישה (מסורתית) זו מול פיתוחים 'לא-קוויים', בהם מחזור החיים מאגד חזרה (איטרציה) משמעותית בתוך התהליך הבסיסי. מחזור החיים הספיראלי הוא דוגמה לגישה לא-קווית טיפוסית.

מטרת חלק 2 היא לקבוע כיצד ניתן ליישם את הקווים המנחים של ISO 9000-3 לתחום פיתוח התוכנה ולספק ייעוץ שימושי בדבר יישום אפקטיבי של מערכות לניהול איכות תוכנה. חלק זה של הספר מחולק לשלושה פרקים:

- פרק 3: ניהול איכות, סיכונים ופרויקטים
- פרק 4: תהליך הפיתוח הבסיסי
- פרק 5: לב מערכת האיכות

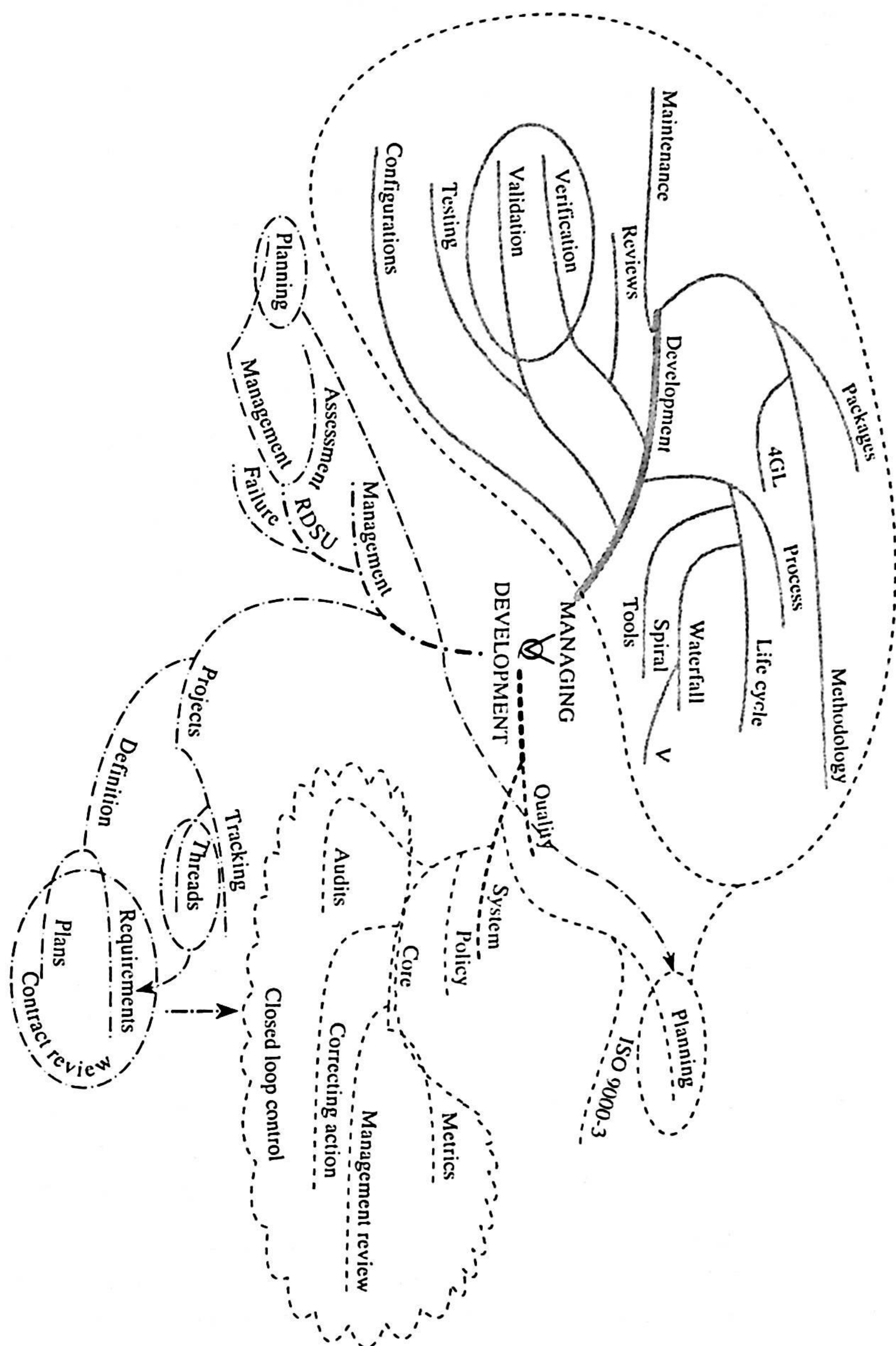
כל פרק בנוי לפי תבנית דומה: תחילה אנו בוחנים את הסוגיות הראשיות שיש לדון בהן; אחר כך מעיינים בקו המנחה של ISO 9000-3; ולבסוף מגישים ייעוץ שימושי בדבר הדרך ליישום אפקטיבי.

פרק 3 שוקל את הסיכונים המאיימים בדרך כלל על פרויקטים של תוכנה ומציג דרכים להימנע מהם, או להתמודד איתם. תכנון האיכות הינו חלק בלתי נפרד מתכנון הפרויקט ומוטבע בדיסציפלינה הכוללת של ניהול פרויקטים.

פרק 4 מתבונן בתהליך פיתוח התוכנה עצמו, ושוקל את הסוגיות העיקריות הכרוכות בהמרת תהליך זה לתהליך הניתן לשליטה. מכאן מוליכה הדרך לדיון מסוים בקטע הגדול יותר של ISO 9000-3.

פרק 5 שוקל את המבנה של מערכת לניהול איכות, כדי לוודא שלא רק הסוגיות שמצאו את ביטוין בפרקים 3 ו- 4 יזכו לטיפול נאות, גם המערכת הכוללת תהיה נתונה לבקרה עצמית, לתחזוקה עצמית ואפילו לשיפור עצמי.

לכשתסיים את חלק 2, אתה אמור להיות כבר בעל ידע מעשי בתקן ISO 9000-3, ולהבין את מה שהקטעים הראשיים שלו מנסים להשיג. אתה אמור גם להיות מסוגל ליישם מערכת לניהול איכות שתעמוד בדרישות לגבי מחזור חיים קווי שגרתי. הגישות הלא-קוויות יידונו בחלק 4 של ספר זה. רעיונות המפתח של חלק 2 מוצגים בתרשים ח.2.1, מפה של תפיסה כללית.



תרשים ח.2.1 דוגמה למפת תפיסה כללית של החלק השני

פרק 3

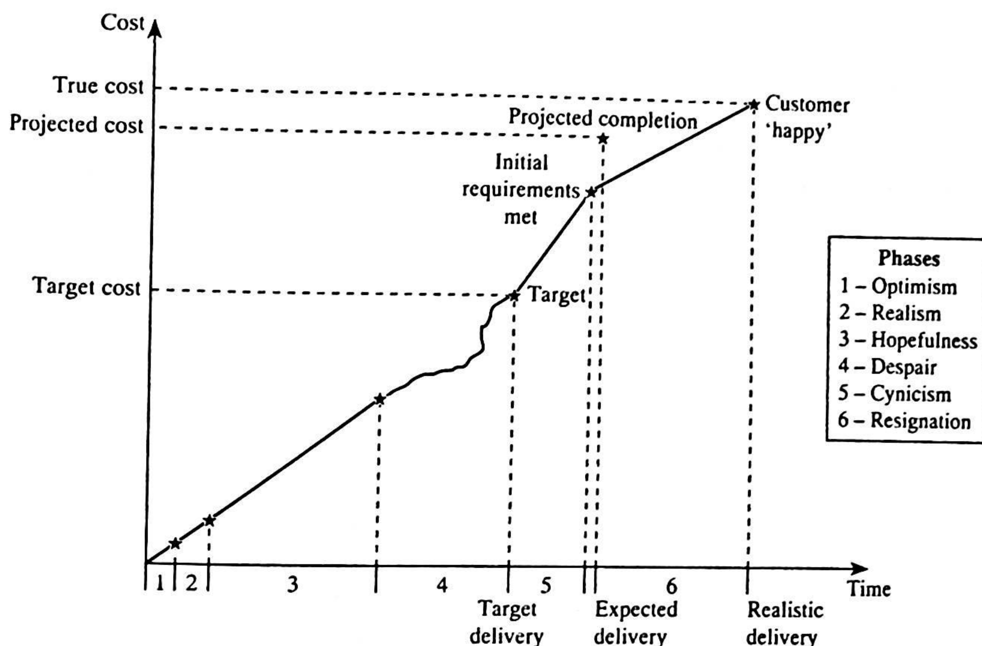
ניהול איכות, סיכונים ופרויקטים

מבוא

מה גורם לכך שפרויקטים של תוכנה משתבשים תמיד? כל כישלון מלמד אותנו דבר מה, אבל נוצר רושם שתמיד מופיעות דרכים חדשות שמוליכות שוב לקראת כשלונות. האם כשל זה הוא בלתי נמנע? האם פרויקטים של תוכנה סתם כך קשים ומורכבים, עד שבני אדם אינם יכולים לשלוט בהם? ייתכן, אך בינתיים טרם הגענו לנקודה בה מסקנה זו תהיה בלתי נמנעת. למען האמת, עלינו להודות, שכמעט לא התחלנו לנצל את הטכניקות העומדות לרשותנו. בסופו של דבר, ההצלחה תבוא כתוצאה מכך שנבין בדיוק את מה שנדרש מאתנו, ומכך שלצורך השגת הדרישות, נשתמש בתהליך מוגדר בבהירות ומובן היטב על ידינו. אם נתמודד עם פיתוח התוכנה בצורה שיטתית, לפחות נוכל להפריד בין הדברים שאנו מכירים לבין אלה שאינם ידועים לנו. את הדברים המוכרים אפשר לעבד לכלל תוכנית פיתוח, והדברים הלא ידועים יאפשרו לנו להעריך את הסיכונים, ולנסח דרכים לצמצומם במסגרת תוכנית איכות. הצעד הראשון לקראת ההצלחה מותנה ביכולת שלנו לשזור יחד שני מרכיבים אלה שבכל פרויקט.

ניתוח הסיבות לכישלון פרויקטים

מה הדבר שמשתבש בפרויקטים התכנות שלנו? כידוע, אין גבול לכושר ההמצאה של האדם, אך עם זאת יש כמה הסברים שכיחים לכישלון. ניתוח אירועים פשוט יוכל לסייע לנו לגבש במקצת את קו המחשבה. תרשים 3.1 ממחיש את אחת השיטות השכיחות יותר שמבטיחות כישלון. הבה נבחן את הפרויקט שלב אחר שלב.



תרשים 3.1 פרויקט תוכנה טיפוסי

שלב 1 - אופטימיות

שלב 1 מתחיל כשמתגלה לקוח שיש לו דרישה הנראית לנו מוכרת במעורפל. ההיכרות היא כזו שמאפשרת לנו לשכנע את עצמנו שיש ביכולתנו לספק את הדרישה במחיר תחרותי ביותר, ובלוח זמנים צפוף למדי. לחילופין, על מחלקת טכנולוגיית המידע (IT) של חברה כלשהי הוטלה המשימה להשיג מערכת מידע ניהולי (MIS) שהיא חיונית להישרדות החברה; האספקה חייבת להיות בלוח זמנים צפוף מאוד 'בגלל סיבות עסקיות קריטיות'. בשלב זה יש רק מעט מאוד מידע מכדי שאפשר יהיה לקיים ניתוח מלא של הדרישות, ונראה שאין מנוס מלקבל החלטה בהקדם שיש להתחיל בפעולה. הדאגות המושמעות על ידי הסגל הטכני אינן זוכות לתשומת לב, משום שאלה תמיד משמיעים קולות דאגה בשלב זה, אך לעולם אין הם יכולים לבסס את דאגותיהם על איזו שהיא טענה מוגדרת. היחס אליהם הוא כאל אנשים המייצרים מראש תירוצים לכישלון העתיד לבוא. מתקבלת החלטה להמשיך במשימה.

שלב 2 - התפכחות

במשך שלב 2 מתחילות להתגלות הדרישות האמיתיות, והערכות גסות מרמזות על מועד אספקה ועלות שיחרגו בכ-50 אחוז מההערכות המקוריות. תחזית זו מבוססת עדיין על מידע לא שלם ותוכניות שהושלמו רק בחלקן, אך היא כוללת כבר מרווח ביטחון נכבד עבור סיכונים שטרם זוהו, והקצאה משמעותית עבור אבטחת איכות

ובדיקות. בהתחשב במידת הערפול של התוכניות, אין ההנהלה משוכנעת שהיעדים ההתחלתיים אינם ניתנים להשגה. בהתחשב בכך שמלכתחילה הקצינו חלק גדול מהמאמץ (קרוב ל-20 אחוז) לאבטחת איכות ולבדיקות, סיכונים בלתי מוגדרים נמחקים מרשימת השיקולים שיש להביא בחשבון.

שלב 3 - תקוות

לאחר מנת היתר של ההתפכחות בשלב השני, מוכן צוות הפרויקט להתארגן לעבודה בסביבה נוחה יותר. כעבור זמן הם שוכחים את כל ההתרגזויות ומתחילים להאמין שאכן יש להם סיכוי להגיע למטרה. התִיכון מתנהל פחות או יותר כשורה, והערכות הזמן והעלויות מתבייתות על היעד הרצוי. מתחילה הצטברות של דחיות פעוטות ובעיות בתִיכון, אך בכוח ההתלהבות מצליחים לשמור אותן בתוך המסגרת, עד למועד שמנהל הפרויקט מתחיל לחשוד שהעבודה שעדיין נותרה לא-בדיוק תתאים למסגרת הזמן העומד לרשותו. מאידך, אין הוא יכול להאיץ את הפרויקט מבלי להגדיל את ההוצאות מעבר לתקציב שממילא כבר מתוח למדי. הוא מתריע בפני ההנהלה.

שלב 4 - ייאוש

ההנהלה ציפתה למתקפת הייאוש של מנהל הפרויקט. הם מרגיעים אותו ואומרים לו שהם עומדים מאחוריו ויש להם אמון ביכולתו להשלים את הפרויקט בזמן ובמסגרת התקציב. יתרה מזאת, צפוי לו פרויקט משמעותי יותר כאשר יסיים את הפרויקט הנוכחי. מובן שאם הפרויקט הזה אמנם יתדרדר, לא כל כך בטוח שצפוי לו פרויקט אחר אחריו.

בינתיים, ביקורות התִיכון ממשיכות לחשוף פגמים עד שמפסיקים את הביקורות כדי לחסוך בזמן. התוכניתנים, שרוצים להפיק יותר קוד ומהר יותר, חוזרים להשתמש בטריקים של תכנות החביבים עליהם. הניפוי נמשך קצת יותר זמן מהצפוי, עד שמגיע הרגע שבו נמסר הקוד למחלקת אבטחת האיכות, מספר ימים לפני המתוכנן. מסתבר שצוות הפרויקט לא הספיק לבדוק את כל התוכנה, אך הזמן והמאמץ שהושקעו בניפוי, מראים את נחרצותם בחיפוש השגיאות. יש עדיין מספר דיווחים על בעיות פתוחות, אך אלו יתבהרו עוד לפני מועד האספקה.

במשך היומיים שהוקצבו לה לבדיקות מגלה מחלקת אבטחת האיכות 20 שגיאות קשות. מתקבלת החלטה לשחרר את התוכנה, בעיקר לאור העובדה שהבעיות הפתוחות כבר טופלו בינתיים (אם כי לא נבדקו במלואן). התוכנה מסופקת, כולל ה'תיקונים' האחרונים שהוכנו על ידי צוות הפרויקט ואשר מחלקת אבטחת האיכות טרם ראתה.

שלב 5 - ציניות

התוכנה נמסרת, אך לא מצליחים להתקינה. כעבור יומיים מצליח צוות ההתקנה, המונה שלושה אנשים, לפתור את הבעיה והמערכת מותקנת בהצלחה. המערכת

'נופלת' מיד לאחר שנמסרה למשתמשים לעבודה. צוות מומחים נשלח למקום ומקדיש שלושה ימים לליבון הבעיה. בהתחשב בתנאים ובדחיפות המצב, לא מעדכנים את התיעוד ומכניסים את ה'תיקונים' לתוך הקוד ישירות. מתחילים בעיבוד חוזר של המוצר, כדי להתגבר על אחדות מהבעיות שהתגלו בימי ההפעלה הראשונים. במשך תקופה של שלושה חודשים, עסוק צוות הפרויקט אך ורק בקיום מצב עבודה של המערכת. מאחר שהם פועלים תחת לחץ לא רגיל, מוותרים להם על משימות לא חיוניות, כגון בקרת התצורה ותיעוד. לאחר חמישה חודשי עבודה, הלקוח מסכים שהתוכנה עונה על דרישות המפרטים המקוריים ומתאימה לפעולה. התוכנה נמסרת רשמית.

שלב 6 - פיטורין

מנהל הפרויקט נתבע להגיש את התפטרותו. רוב חברי צוות הפרויקט מתפטרים כדי לעבור לבית תוכנה אחר, שבו הם מאמינים, המצב יהיה שונה. צוות התחזוקה משלים עם הצורך לצאת למסלול כבד וממושך: הטמעת אוסף הדרישות לשינויים בתוך התוכנה הלא-מתועדת, והתצורה הלא-ידועה המבוססת על תִיכון 'מקוצץ'. כדי לפצות על המאמץ הנוסף שהושקע בפרויקט זה, מוצא מנהל המכירות פרויקט אחר שניתן לטפל בו בקלות, כהרחבה קלה יחסית של הפרויקט הקודם ומסכם על מחיר טוב המבוסס על אספקה מהירה, וחוזר חלילה.

מה השתבש כאן

החברה הדמיונית באירוע שלנו, הכניסה את עצמה למעגל קסמים מוכר היטב. לדאבוננו, התנאים המוליכים למעגל זה שכיחים למדי. לעיתים תכופות הלחצים המסחריים מאלצים אותנו להיכנס למצבים שבהם אנו מעדיפים לקבל את אפשרות הכישלון בתור המחיר שנשלם עבור הצלחה אפשרית בעתיד.

מעגל הקסמים מתהדק באמת כשאנו מעדיפים להתעלם מהבעיות, או מעמידים פנים כאילו אנו שולטים במצב.

כיצד נמנעים מסיכונים ממין זה? על ידי הקדשת תשומת לב נאותה לתהליכי ניהול פרויקטים ופיתוח תוכנה. הבעיות שזוהו בעת ניתוח פרויקט שנכשל ישמשו כבסיס התחלתי ליצירת קבוצת דרישות בסיסית לתהליך ניהול פרויקט, שאליה נוכל להוסיף עכשיו קטע בדבר ניהול סיכונים.

עשרה סיכונים מיותרים בפרויקט תוכנה

---

1. אין בידינו ידע מספיק על עסק הלקוח, שיאפשר לנו להבין את ציפיותיו במלואן.
2. איננו מנתחים ומתעדים את הדרישות די הצורך.
3. איננו מקיימים בקרה רשמית מספקת על השינויים בדרישות.

4. איננו מתכננים תכנון מציאותי. אנו מרשים לעצמנו להיות מונעים על ידי האירועים.
5. כאשר אנו נמצאים תחת לחץ, איננו נצמדים לדיסציפלינות שקבענו לעצמנו.
6. איננו מזהים את כל הבעיות הצפויות ואיננו מטפלים בהן.
7. איננו מקילים על הגשת דיווחים על בעיות.
8. איננו מקדישים תשומת לב מספקת לסימנים המעידים על בעיות.
9. איננו מגדירים באופן ברור את תחומי האחריות.
10. איננו דואגים לעדכן את כל חברי הצוות בהתקדמות הפרויקט.

במילים פשוטות, איננו מעריכים את כל הסיכונים הנלווים לפרויקט, ואיננו מתכננים כיצד להימנע מהם או לצמצם.

ניהול סיכונים

פיתוח תוכנה, בדומה לכל פעילות יצירתית, מכיל בחובו וודאות של עשיית שגיאות. שגיאות המתגלות, מוליכות בדרך כלל, לחזרה מסוימת על פעילויות כדי לסלקן. עבודה נוספת זו, שנגרמת עקב תקלות, גורמת לאיחור בביצוע הפרויקט, אלא אם תוכננה מראש אפשרות זו והוקצה זמן לעבודות חוזרות. במקרה שהפרויקט מבוסס על מועד אספקה מוגדר, או שהוא מוגבל לתקציב מסוים, העבודה החוזרת עלולה לאיים על עמידה מוצלחת ביעדי הפרויקט (שגיאות שלא נגלה, יועברו בירושה ללקוחות ולמשתמשים, ואלה לא יאחרו לגמול לנו על כך).

כדי למנוע מצב של חריגה בתקציב וגלישה בלוח הזמנים צריך לפעול להערכת הסיכונים וטיפול בחריגות. זהו **ניהול סיכונים** (Risk management).

עבודה לא מתוכננת (כלומר, מטלות שיצורפו במהלך הפיתוח, ללא תכנון מוקדם) משמשת דוגמה לסוג הסיכון המשותף לרוב הפרויקטים של פיתוח תוכנה; סיכון זה הוא אחד הרכיבים המציאותיים של התכנון שהזכרנו בסעיף הקודם. צפוי שיופיעו גם סיכונים אחרים, רבים מסיכונים אלה קיימים, במידה קטנה או גדולה, בכל הפרויקטים. מהם סיכונים אלה?

סיכונים שכיחים שיש להישמר מפניהם

סיכונים שמקורם בלקוח

- דרישות בלתי ידועות, או לא מובנות במלואן;
- דרישות מעורפלות, אך ציפיות גבוהות;
- העדר תחושת בעלות: האחריות מושארת למפתחים;

- פיגורים בהגדרת מידע קריטי;
- שינויים תכופים של הדרישות;
- מחיר המבוסס על ניתוח לא שלם של הדרישות.

סיכונים שמקורם במפתחים

- הגדרה לא מספקת של מוצרים מוגמרים;
- יותר מדי חדשנות בדרישות. ניסיון להטמיע יותר מדי תפיסות חדשות;
- עובדים שאינם די מאומנים ומנוסים לביצוע הפרויקט;
- צורך בכלים ו/או בטכניקות מיוחדים שאינם זמינים כרגע;
- שיטות ו/או כלי פיתוח שאינם הולמים את המשימה הנוכחית.

סיכונים שמקורם בתכנון

- לא קיים תכנון לאימות ולבדיקת התקפות;
- לא הוקצו משאבים עבור עבודה חוזרת (rework);
- לא הוקצה די זמן לפיתוח;
- התוכניות מבוססות על ציפיות הלקוח, ולא על יכולת המפתחים לספק את המוצר, או המוצרים;
- לא הוקצה די זמן לבדיקות.

יש צורך לסלק, או לטפל בכל מה שיכול לאיים על הפרויקט, כדי לוודא שמשעה שנתחיל בפעילויות הפיתוח, נוכל לעמוד בהצלחה ביעדים. ייתכן שאופי האיומים האלה ימנע את האפשרות להבטיח סיום מוצלח, וייתכן שעלות סילוקם של כל הסיכונים תהיה גבוהה ביותר. לעומת זאת, היתרונות הצפויים מהצלחה מלאה של פרויקט מצדיקים הוצאה מסוימת על צעדים שמטרתם צמצום הסיכונים.

עלינו לבחור באסטרטגיה כזאת:

- שתזהה, תכמת ותתריע בפני כולם על הסיכונים שאנחנו צופים שיהיו;
- שתסלק או תצמצם בצורה משמעותית את הסיכונים שמהווים את האיום הגדול ביותר;
- שתאפשר לנו לבקר ולשלוט בסיכונים הנותרים;
- שתקיים לעיתים מזומנות סקירת מצב הסיכונים.

אסטרטגיית ניהול סיכונים מעשית תזהה את כל הסיכונים המהווים את האיום החמור ביותר על הפרויקט. לגבי כל סיכון, עלינו לתכנן צמצום אפשרות הופעתו, או צמצום השפעתו עד לרמה נסבלת, במקרה ולא ניתן למנוע אותו כליל. הבחירה בין הימנעות מוחלטת מסיכונים, לבין הסתייעות בתוכניות לשעת חירום לטיפול בסיכונים אלה, מותנית בממדים ובאופי של כל סיכון וסיכון. מה שאסור לעשות הוא **להתעלם מסיכונים**, או לקוות שהם פשוט לא יקרו. יש מקום להחלטה שלא ניתן לסלק או

לטפל בדרך נאותה בסיכון מסוים. עם זאת, עלינו להיות מודעים לקיומו, כדי שנוכל לעקוב אחר המצב ולקבל החלטות מושכלות על הדרך בה ננהג במקרה שיחולו שינויים בנסיבות.

יש לצפות שניהול הסיכונים ינקז חלק ממשאבי הפרויקט לכיוון של נקיטת צעדים הגנתיים: הדבר עשוי להקטין את הפריון, להגדיל עלויות או לחייב את צמצום הדרישות. האסטרטגיה שנאמץ, חייבת להיות גם קבילה וגם כזאת שתוכל **להגדיל במידה משמעותית את סיכויי ההצלחה של הפרויקט** שיותאם לנסיבות.

צמצום הסיכונים - תכנון האיכות

תכנון האיכות הוא 'הצד השני' של תכנון הפיתוח. בשעה שתוכנית הפיתוח יוצרת מתודולוגיה לסיום מוצלח של הפרויקט, תוכנית האיכות מחפשת לסלק, או לקזז את כל ההשפעות הפועלות **בניגוד** להשלמתו המוצלחת.

ניתוח הסיכונים המתבצע כחלק מתכנון הפיתוח אמור לזהות כל איום אפשרי לסיום מוצלח של הפרויקט. תוכנית האיכות חייבת להיכנס לפעולה נגד כל אחד מהאיומים האלה.

במקרים רבים, האיום הראשי על כל פרויקט, הוא האיום של תוהו ובוהו ואי-סדר. אי ודאות בדבר דרישות הלקוח או של יעדי האיכות, עלולה להוליך לדיסהרמוניה, בעוד אחדים מעובדי הצוות ממשיכים לשקוד בחריצות על השגת מה שנראה להם כיעדים הלגיטימיים של הפרויקט. בדיקות חיוניות ופעולות ביקורת עשויות שלא להתבצע, משום שלא נקבעה האחריות לביצועם. דברים אלה נכונים במיוחד במצבים בהם נתון הפרויקט בלחץ, וביצוע בדיקות וסקירות מייצגים מחסומים בדרך למסירתו של קטע תוכנה שאספקתו התאחרה.

משום כך, **המשימה הראשונה של תוכנית איכות היא לוודא את זמינות הדרישות, יעדים, ותחומי אחריות.** משימה זו צריכה לכלול את הצורך לשקול בקפדנות את האסטרטגיה של האימות ובדיקת התקפות, תוך הבאה בחשבון של הסיכונים הטכניים המזוהים. תוכנית בדיקות הנבנית בשלב מוקדם זה, יכולה לאפשר התייחסות לכל הסוגיות הקריטיות לפני שיחל המרוץ נגד הזמן, עליה להגדיר גישה לנושא הבדיקות שתגדיל את הסיכויים לאספקת מוצרים באיכות מניחה את הדעת.

תוכנית האיכות מונעת תוהו ובוהו ואי-סדר משום שהיא מוודאת שכל מקרה, צפוי או בלתי צפוי, יימצא בתחום האחריות של מישהו מאנשי הצוות.

קיימות דרכים למדד בדיקות, כגון יחס של כל משפטי הקוד שעברו כבר את הבדיקות, או מספר הפונקציות שצריך לבדוק בנפרד. יש באלה כדי לסייע לכימות רמת הבדיקות הנדרשת. התוכנית שהוכנה מראש היא הדבר הראשון שאנו יכולים להסתמך עליו, במקרה שבמועד מאוחר יותר מתחיל הלחץ להוות בעיה משמעותית. במקרה שהלחץ נעשה כה גדול, עד כדי פיתוי לנטילת סיכונים, כגון צמצום מספר הבדיקות המתבצעות, נוכל להסתייע במדידות אלו שתאפשרנה לנו לכמת את גודל הסיכון שאנו נוטלים על עצמנו.

עובד המבצע את הבדיקות ומגיב על הדרישה לקיצוצים בתלונה בלבד שלא ניתן לו מספיק זמן לצורך זה, אינו מספק למנהל הפרויקט נתונים בדבר מידת החוכמה או האיוולת שבקיצוצים אלה. לעומתו, מבצע בדיקות המציין שהושלמו 57 אחוז מהבדיקות הפונקציונליות המתוכננות, אך אלה כללו רק 49 אחוז של הקוד, וכי במשאבים הקיימים ניתן יהיה להשלים את שאר הבדיקות תוך 17 יום, או תוך 12 יום במקרה שיוקצה לו עובד נוסף, יזכה למלוא ההקשבה והפתיחות של מנהל הפרויקט על האסטרטגיה היעילה ביותר שיש לנקוט בשלב זה של הפרויקט.

האיום השני המשותף לכל הפרויקטים הוא ההשפעה של שינויים בלתי מבוקרים. שינויים יחולו תמיד בכל פרויקט תוכנה. תוך כדי התקדמות הפרויקט צפויות להתגלות דרישות שאינן שלמות, או שאינן נכונות; עלולים לחול שינויים בהבנת האופי האמיתי של דרישות המזמין; שגיאות שהתגלו בזמן הפיתוח, מתוקנות וגורמות לפעמים ליצירת שגיאות חדשות. כל אלה מהווים מרשם לשינויים תכופים ומפליגים. בהעדר בקרה קפדנית, אנו עלולים לגלות שחלקים שונים של המערכת אינם תואמים עוד, או שחלקים של המוצר המוגמר אינם תואמים את הדרישות הנוכחיות. משעה שנפגמת יכולת המעקב והקישור בין הדרישות לבין המוצרים המוגמרים, תהיה ההתאוששות קשה יותר.

משימתה השנייה של תוכנית איכות תהיה אם כן, לוודא שכל שינוי יהיה מבוקר, בין אם מקורו בפגמים, ובין אם מקורו בדרישות חדשות. כדי לעשות זאת תצטרך התוכנית להגדיר את הכללים ואת תחומי האחריות לגבי שינויים מכל הסוגים.

איום שלישי נובע מהחדשנות ומהמורכבות. אם הפרויקט מכיל אלמנטים חדשים עבורנו או מורכבים מאלה שעסקנו בהם לפני כן, יהיה עלינו לנטר את הפרויקט ביתר זהירות ולעיתים תכופות יותר, כדי לוודא שאיננו סוטים מהמסלול. פעילויות אימות נוספות, כגון סקירות ובדיקות, עשויים להידרש כדי לספק את נקודת המבט הנדרשת, ואת אלה יש לכלול בלוח הזמנים של התוכנית.

משימתה השלישית של תוכנית איכות היא לוודא שהתוכנית תהיה מציאותית ככל האפשר. חייב להיות ביטחון מלא שניתן להשלים את הפרויקט בהצלחה, במסגרת הזמן והמשאבים שהוקצבו בתוכנית הפיתוח. אחרת יש חשש שתוכנית הפיתוח תאבד במהרה את אמינותה ויתעלמו ממנה.

משימה נוספת, אחרונה, של תוכנית האיכות היא תיעוד כל סיכון אחר שלא נדון קודם לכן וזיהוי אסטרטגיית ניהול הסיכון הנדרשת, או מניעתו. ייתכן שנחליט לא להכין תוכנית פעולה לגבי כל אחד מהסיכונים, אך עם זאת, עלינו לנהל מעקב אחר כל הסיכונים בהם החלטנו לא לטפל, אחרת הם עלולים לבצבץ ולפגוע בעיתויים הפחות צפויים. **שימת מחסום לסיכונים היא מרכיב יסוד של תוכנית איכות.**

מה צריכה להכיל תוכנית איכות טובה

1. **כללים ותחומי אחריות**, בעיקר אלה הכוללים אחריות וסמכות, כגון:
   - ◊ סמכות להגדרת דרישות.
   - ◊ סמכות לביצוע תִיכּוּן.
   - ◊ סמכות להרצת בדיקות ולשילוב (integration).
2. **בדיקות**

ייתכן שתהיה תוכנית בדיקה נפרדת עבור פרויקטים גדולים יותר, אך צריך לנסח כאן את אסטרטגיית הבדיקות הכוללת, בתוספת המדדים שישמשו לקביעה מתי אפשר לקבוע שהמוצר נבדק די הצורך.

3. **ניהול התצורה**

ייתכן שתהיה התייחסות לתוכנית נפרדת. יש צורך להגדיר את המנגנון הבסיסי של זיהוי ויכולת המעקב, וגם את נהלי בקרת השינויים.

4. **דיווח בעיות ותקלות**

יש להגדיר לכל העובדים את השיטות בהן יש להשתמש כדי לדווח על בעיות ותקלות שונות.

5. **מדדים**

צריך להגדיר את כל המדדים שישתמשו בהם עבור המוצר או תהליכי הפיתוח. יש להסביר לכל העובדים בפירוט מספיק את איסוף הנתונים הדרוש, כדי לוודא שהבינו את שנדרש מהם.

6. **אישורים**

יש לקבוע את האנשים שתוענק להם הסמכות לאשר את כל מסמכי הפרויקט.

7. **סיכונים**

לצורך ידיעה והנחייה של העובדים, יש לכלול רשימה של כל הסיכונים הכרוכים בפרויקט ושל אסטרטגיות הניהול שיש לנקוט. כך אפשר לוודא שהעובדים לא ייכנסו לתחומי סיכון לא מתועדים, שאין עבורם תוכנית מילוט מתאימה.

8. **יעדי איכות**

יש להגדיר את היעדים שנקבעו למוצרים המוגמרים, יחד עם הגדרות של דרישות איכות כלליות, כגון רמת הבדיקות המזערית הקבילה לגבי הפרויקט.

9. **קריטריונים ותוכניות למבחני קבלה**

יש להגדיר את הקריטריונים שנקבעו למבחני הקבלה, ולכלול התייחסות לתוכניות הבדיקה של מבחני הקבלה.

10. **תאריכי מפתח**

יש לזהות ולהגדיר את אבני הדרך הראשיות.

11. **סקירות חוזרות**

יש להגדיר את המדיניות לגבי סקירות חוזרות של המסמכים והקוד, בנוסף לכללי התנהגות.

12. **הרצות נסיוניות על ידי המשתמש**

בכל מקרה שהפרויקט כולל הרצות נסיוניות על ידי המשתמש, יש להגדיר את הדרישות ואת התוכניות שהוגדרו או אוזכרו לצורך זה.

13. **כללי עיצוב וקידוד**

יש להגדיר או לאזכר את הכללים והנהלים עבור כל פעילויות התִיכוּן והקידוד.

הבהרה מלאה של הדרישות

עד כה עסקנו רק בניהול הפרויקט, אך לא בהגדרתו. דיסציפלינת הניהול יכולה להיות מעולה ככל שתהיה, לעולם לא תוכל להצליח, אלא אם יהיה המאמץ מכוון כהלכה לקראת היעד. זו מטרת **ניתוח הדרישות**.

עלינו להכין תיאור בהיר ושלם של הציפיות הלגיטימיות של הלקוח, הפונקציונליות והבלתי פונקציונליות, ולתעד אותן בצורה תמציתית, בהירה, ועקבית, רק כך הלקוח יוכל לאשרן ולפעול על פיהן.

> ***אנחנו חייבים להאזין ללקוחות ולהשתמש בשפתם. לדוגמה, הגדרת המילה 'באיחור' יכולה להיות בעלת חשיבות עילאית.***
>
> ***קאמרון לאו (SQM, 1993)***

הגדרת תפקוד המוצר

יש דרכים רבות להגדרת הדרישות. הכלים המשמשים לכך יכולים לכלול, החל מניסוח מילולי בלתי מובנה, עד לשפת מפרטים רשמית, או ממסמכים שבכתב עד לאבטיפוס דינמי. קיימים כמה קריטריונים חיוניים להשלמת פעולת ניתוח הדרישות, מבלי להתייחס לדרך בה מתבצעת המשימה. קריטריונים אלה כוללים:

- קיום רישום קבוע כלשהו, אשר יכיל את הדרישות המוסכמות, ושישמש בסיס להסכמה על מה שצריך ולא צריך להיעשות;
- מזהה ייחודי של גרסת הדרישות המוסכמת, כדי שאפשר יהיה לעקוב אחר שינויים בדרישות על ידי יצירת סדרת גרסאות חדשות, המייצגות מצב חדש של הדרישות;
- קבוצת קריטריונים לקבלת המוצר, שישמשו בסיס להסכמה על מה שנחשב כקביל בשלב המסירה.

המשימה הבאה, לאחר קביעת קו הבסיס הרשמי, היא ארגון הדרישות להקלת התיכון, היישום והסקירה. המפתחים צריכים להיות מסוגלים לוודא, שכל הדרישות זכו לטיפול בכל אחד משלבי הפיתוח, ושלא נעשתה, או שלא תיעשה כל עבודה בהיבטים שאינם כלולים בקו הבסיס הרשמי.

המפתחים צריכים לדעת את החשיבות היחסית של הדרישות השונות, כדי שאפשר יהיה לתכנן את הבדיקות. הבדיקות הן פעילות מדגמית, וצוות הבדיקה ירצה להיות בטוח שהבדיקות המתוכננות מתמקדות בתחומי הדרישות בעלות העדיפות הגבוהה יותר.

יש גם ערך רב ליכולת לזהות אילו חלקים של הדרישות מיושמים על ידי איזה חלקים של המערכת. למידע זה תהיה חשיבות מיוחדת במועד מאוחר יותר, כאשר יהיה צורך לשקול את השפעת השינויים לפני מתן אישור לשינוי כלשהו.

הגדרת דרישות האיכות

השגת האיכות מותנית בקיום **הגדרה הולמת של תכונות האיכות** ובידיעת מימדי כל תכונה. בהקשר זה המילה 'הולמת' פירושה **אובייקטיבית, ניתנת לכימות ולמדידה**. במציאות, רק לעיתים רחוקות אפשר להגיע להגדרה סגורה שאינה מותירה מקום לספקות. המפתחים נאלצים להתפשר לעיתים תכופות על דרישות איכות סובייקטיביות שצורת הצגתן אינה תמיד כה מגובשת.

השגת איכות התוכנה היא אם כן פחות ממאמץ מדעי מושלם. הגברת השימוש בשיטות דיסציפלינריות ומובנות, ותכנון זהיר, לצמצום השפעת סיכונים ידועים, אינה אלא תחליף בלתי מושלם לתהליך שחייב, בדרך כלל, לספק בדיוק את הנדרש. עם זאת, זו הדרך הטובה ביותר העומדת לרשותנו בשלב זה. מתוך ידיעת מגבלות אלו, חשוב מאוד לשאוף להבנה אפשרית טובה ביותר של הדרישות, לפני שנתחיל לשקול את האסטרטגיות בהן נוכל להשתמש להשגת דרישות אלו.

'מוצר איכותי': מוצר המבצע את רוב הדברים שאנו רוצים שיבצע, ואשר יש ביכולתנו להשלים עם פגמיו.

פאול הרצליך, Systéme Evolutif (SQM, 1993)

כמה מתכונות המפתח האיכותיות עשויות לפעמים להשתמע ולא להיאמר במפורש. כאשר תכונות אלו הן ביסוד תרבות המשתמש, ייתכן ולא ניתן יהיה לזהותן בצורה מוגדרת. לעובדת היותן מעין 'טבע שני' בקהילת המשתמש, יש כמעט משום ערובה שהמפתחים לא יבינו אותן במלואן. מאידך, ייתכן שקהילת המשתמש לא תהיה מסוגלת להבין בעצמה את דרישותיה במידה שתאפשר לה להכיר את כל השלכותיהן האיכותיות. בכל מקרה, נטל הזיהוי, כימות ותיעוד הדרישות הבלתי פונקציונליות חל על המפתחים, ויש להביאו בחשבון בעת ניתוח הדרישות.

לאחר זיהוי תכונות האיכות, חייב המפתח לוודא שהלקוח מסכים עם מה שתועד, וכי יש לו ידיעה ברורה על הדרך בה יתנהג המוצר שיתקבל. כך אפשר יהיה להגדיר מבחני קבלה מתאימים שימדדו את המוצר מול מדד (קריטריון) כלשהו של התאמה, אשר יישקף את ציפיות המשתמש. קבוצת דרישות שסוכם עליה רשמית, חיונית להגדרת המדדים לקבלת המוצר, ולמבחני הקבלה שישמשו בסופו של דבר, בסיס להסכמה בדבר קבילות המערכת. המפתחים ישתמשו בדרישות מוסכמות אלו כבסיס לתִיכון, כך שיהיה להם הביטחון שהם אכן יוצרים מוצר העונה על ציפיות המשתמשים. גם במקרה שבשלב מוקדם זה, הדרישות מובנות רק באופן חלקי, חובה לקבוע רשמית קו-בסיס ידוע ומוכר, ממנו ניתן להמשיך בפעילות בביטחון.

מאוחר יותר, אם וכאשר יחולו שינויים בדרישות, חייבת להיות אפשרות לעקוב בהצלחה אחר השלכות השינויים האלה במהלך עבודת הפיתוח המוגמרת, כדי לוודא שלא נתעלם מאחת מהן. בדומה, יש צורך לאפשר מעקב אחורה, אחר שינויים בתִיכון, כדי לקבוע אם ומהן ההשפעות שיש לשינויים אלה על הדרישות.

מעקב ובקרה של שינויים הם היבט חשוב מאוד של ניהול התצורה, וברור שיש צורך במערכת אפקטיבית לניהול התצורה כבר מהתחלת כל תרגיל בפיתוח תוכנה. מתודולוגיה מוגדרת היטב חשובה באותה מידה, כך שאפשר יהיה לעקוב אחר התִיכון, ולהעריך בדייקנות ובמהירות את השלכות השינויים.

מדידת איכות המוצר

כדי שנוכל לקבוע את המועד בו המוצר יהיה מוכן למסירה, ראוי להגדיר מדדים של התכונות הצפויות שלו. גישה זו מהווה שיפור רב של מדיניות בדיקות הנמשכות כל עוד אנו יכולים להרשות זאת לעצמנו, ואז לקוות שכשל המוצר לא יהיה דרמטי יותר מדי לאחר המעבר להרצה! רוב המפתחים נוהגים כאילו הם מצפים למשוב שלילי כלשהו מלקוחותיהם, דבר המצביע שהם עצמם סבורים שהאסטרטגיה שלהם היא פחות ממושלמת. קבוצה של תכונות מכומתות, יכולה לספק למפתחים יעד ברור לפעולה, ואף תיתן להם קבוצת מדדים חד-משמעית לבדיקת המוצר.

לדאבוננו, כאשר עוסקים במוצרי תוכנה, קשה מאוד להגדיר תכונות הניתנות למדידה. אם לא די בכך, לא קיימת תרבות מדידה בענף התוכנה, ואלה המבקשים לכלול את המדידה בתהליך הפיתוח, נתקלים לעיתים תכופות בהתנגדות חריפה מצד ההנהלה, העובדים, וגם הלקוחות. למרות שבעיקרון כולם מסכימים שמדידה היא דבר טוב כשלעצמו, מעטים מוכנים להוציא כסף לצורך זה.

אחת הבעיות שבהנהגת תרבות מדידה, מקורה בחוסר יכולת להשוות בין המדדים המעטים שנבדקו על ידי חברות שונות. כל מאמץ מדידה דומה לאי המנותק מכל פעילות מדידה אחרת. כל עוד פעילות המדידה נעשית בקנה מידה קטן, אין תמריץ לשיתוף מידע, אפילו בהגדרות המדידה. בפרק 6 נעסוק בסוגיה זו ביתר הרחבה.

דרישות המעקב

אחת הבעיות הניצבות בפני המפתחים היא כיצד לוודא שכל הדרישות מטופלות בצורה נכונה תוך כדי תהליך התיכון, כלומר, התיכון **עונה על כל הדרישות**. לעומת זאת, בעיה לא פחות חשובה היא היכולת לוודא **שרק** הדרישות בלבד מטופלות על ידי התיכון - שהתיכון הוא מינימלי, ואינו גולש מעבר לדרישות עצמן. נושא זה יכול להיות חשוב בעיקר במצב של התקשרות חוזית במחיר קבוע, כאשר ה'תוספות' יכולות לצרוך זמן ומשאבים שלא נדונו בהצעה המקורית.

ככל שהפיתוח מסתעף לתחומים מפורטים יותר ויותר, יש צורך בשיטה שתבטיח עבודה לאור הדרישות. בכל שלב של התהליך נזדקק לדרך פשוטה שתאפשר לקבוע אם אמנם ענינו על הקריטריונים החיוניים של שלמות ומינימליות.

סקר החוזה

הגדרה דקדקנית ככל האפשר של הדרישות אינה המשימה היחידה שעלינו לבצע לפני התחלת הפיתוח. עלינו לוודא שהעבודה תוכל להסתיים לשביעות רצוננו ורצון הלקוחות גם יחד. עבודה המתבצעת עבור לקוח ועל פי חוזה, מחייבת כמובן הגדרה כלשהי של הסיום המוצלח, שישמש תנאי לתשלום. כיצד נוודא שלא נתקשר בחוזים שאין ביכולתנו לעמוד בהם בהצלחה? כיצד יכולים הלקוחות להתגונן מפני האפשרות שנבטיח להם יותר ממה שנוכל לספק, ובלבד שנזכה בחוזה?

מטרת הסקר החוזה היא יצירת בטחונות עבור שני הצדדים על ידי התייחסות לשאלות מפתח, עוד לפני שהצדדים מקבלים על עצמם התחייבויות חוזיות. הדרישות והקריטריונים לקבלת המוצר, עם כל חשיבותם, אינם הסוגיה היחידה. לוחות הזמנים לאספקה יכולים להיות קריטיים ללקוח. יש צורך לסכם גם על שיטת ומקום האספקה. התקנת תוכנה בתחנת כוח אטומית בסיביר, אינה דומה לאספקת כמה תקליטונים באמצעות הדואר. הניסיון מלמד על מספר לא מבוטל של מלכודות הממתינות לאלה שאינם נזהרים, והמְפַתח החכם ייטיב לעשות אם יפעל לפי רשימה מוכנה מראש של סוגיות שיש לטפל בהן לפני חתימת החוזה. גם ללקוחות של מפַתחי התוכנה יש רשימות משלהם. ברור אפוא, שחשוב מאוד ששני הצדדים ישקלו את כל הסוגיות שלפניהם, ואף ידונו ויסכמו אותן ביניהם.

חשוב לזכור שחוזה נאה ומסודר, המכיל את כל הדרישות שסוכם עליהן בעת הסקר שלו, הוא חריג ולא מצב מקובל. בדרך כלל, נמשכים הדיונים בדרישות ובמדדים לקבלת המוצר, זמן רב לאחר חתימת החוזים. במקרים אלה אנו זקוקים לדרך שתאפשר מציאת פתרון לקשיים שעלולים להתעורר, בעת הגדרת הדרישות ,או מאוחר יותר, בעת ביצוע החוזה. היבט חשוב אחד שיש לסכם בעת סקר החוזה הוא המנגנון שבאמצעותו ניתן יהיה למצוא פתרון למחלוקות בנושאים אלה. שמות אנשי קשר שישמשו לתקשורת בין הצדדים, ונהלים להעלאת סוגיות חוזיות ופתרונן. אלה יכולים להיות חלק משגרת הסכמות שבזמן חתימת החוזה. לעומת זאת, קשה יהיה ללבן דברים אלה במקרה ויתעוררו סוגיות חוזיות במועד מאוחר יותר.

מספר שאלות שצריכות להישאל בעת סקר החוזה

1. האם אנו יודעים מהן הדרישות, ומבינים אותן?
2. האם הדרישות כתובות ומנוסחות בבהירות, ומזוהות בצורה נאותה?
3. האם אנו מצפים לבעיות טכניות כלשהן?
4. האם אנו יכולים לעמוד בלוח הזמנים?
5. האם יש לרשותנו די משאבים להשלמת הפרויקט בזמן?
6. האם אנו זקוקים לכלים ו/או לטכניקות מיוחדות כלשהם?
7. האם בוצע ניתוח סיכונים?
8. האם מינינו איש קשר למגעים עם הלקוח?
9. האם בוצעו או תוכננו סקרים במשותף עם הלקוח?

כיצד מטפל תקן ISO 9000-3 בסוגיות אלו

אחד המשפטים החשובים ביותר בתקן ISO 9000-3 כלול בסעיף תמים למראה (5.1) שנושא את הכותרת 'כללי'. ההנחיה המופיעה בסעיף 5.1 אומרת בפשטות, שפרויקטים של פיתוח תוכנה צריכים להיבנות על פי **מודל מחזור החיים**.

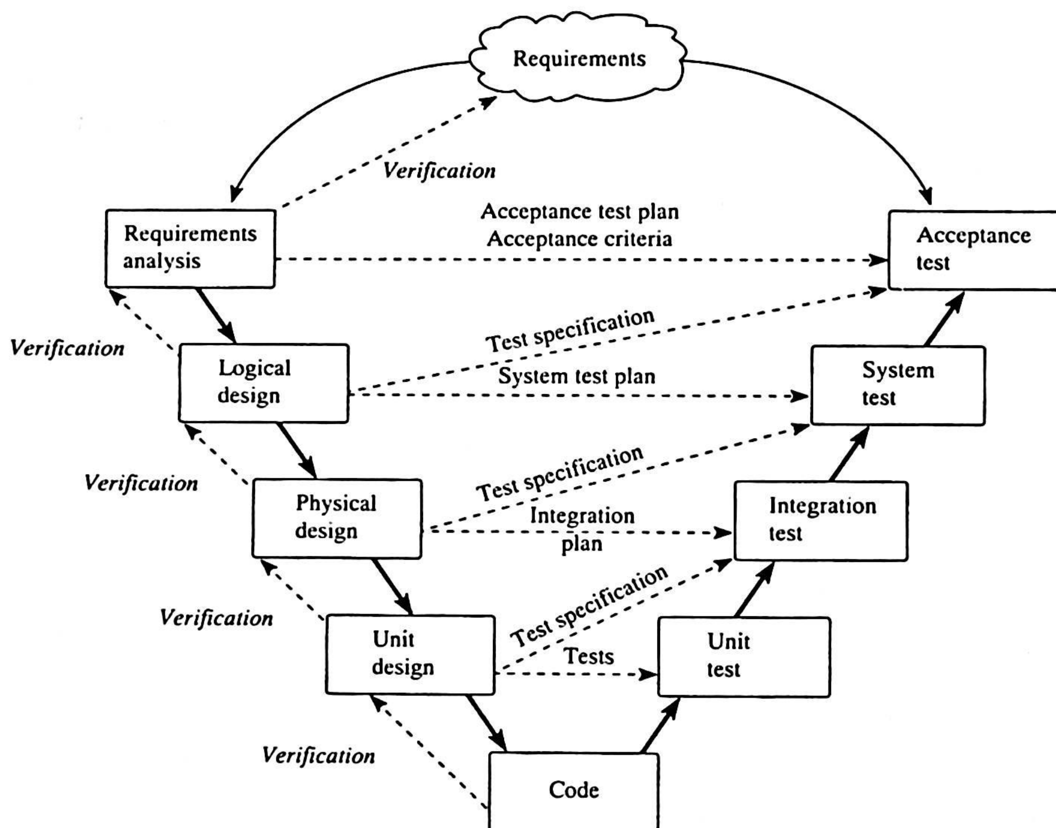
השאלה של מחזור החיים

הקווים המנחים נמנעים במפורש מהתייחסות למודל כלשהו של מחזור החיים, אך מתאר הגישה הכללית יתאים למדי למודל מסוג מחזור החיים מטיפוס V שמתואר בתרשים 3.2. פירוט מודל זה אינו מעניננו ברגע זה; חשוב רק שנוכל לזהות את השלבים, הקשרים, והיחסים שביניהם. במקרה זה קיים רצף של תִיכון ראשוני, אך יש גם קישורים בין פעילויות התִיכון והבדיקה. הצורה הכללית ומבנה המודל מספקים את המסגרת, לאו דווקא את הפירוט הספציפי של השלבים, אך כאשר הקלט והפלט של השלבים השונים מוגדרים בפירוט רב יותר, יסייע הדבר בניסוח מתודולוגיית הפיתוח. אז ניתן יהיה להקים סביב למתודולוגיה, את מבני תוכניות הפרויקטים.

תכנון פרויקטים

תכנון פרויקטים מכיל מספר רב של רכיבים. בדומה לתוכניות הפיתוח, המזהות וקובעות לוחות זמנים לפעילויות הפיתוח היצירתיות, יכולות גם תוכניות איכות, תוכניות לניהול התצורה, תוכניות לבדיקות, תוכניות למבחני קבלה על ידי המשתמשים, תוכניות יישום, תוכניות תחזוקה, ועוד הרבה אפשרויות אחרות. תקן ISO 9000-3 מציין במפורש תוכניות פיתוח, תוכניות איכות, תוכניות לניהול התצורה ותוכניות תחזוקה, אך אין לצפות שכל אחת מאלו תצדיק בהכרח יצירת מסמך ספציפי. במספר מקרים, יכולים כל ההיבטים האלה להיכלל **במסמך יחיד של תכנון**

פרויקטים. המדד העיקרי הוא, האם ניתן במקום כלשהו, ביטוי נאות לסוגיות שהועלו על ידי תקן ISO 9000-3. בפרק זה נדון בהיבטים העיקריים של תכנון הפיתוח ותכנון האיכות.



תרשים 3.2 מחזור חיים מטיפוס V

תכנון הפיתוח

משעה שפותח מודל מחזור החיים, הופכת פעולת תכנון הפיתוח לפעולה פשוטה יחסית. משימת תוכנית הפיתוח היא, לקחת מודל כללי וליצור ממנו מודל מוגדר שיאומץ להשגת המוצר הנדרש. דבר זה יכול להתבטא בפשטות בהפעלת המודל של מחזור החיים בשלמותו, או שייתכן שיהיה צורך להמציא מודל חדש לגמרי עבור הפיתוח המסוים.

בכל מקרה נזדקק ל'מפת דרכים' מפורטת שתזהה את אשר יש לעשות, על ידי מי ומתי. הפעילויות שבתוכנית ינבעו מהשלבים של מחזור החיים תוך כדי יישומו לבעיה שבטיפול. המוצרים הסופיים של השלב חייבים להיות מגובשים וספציפיים ולא

כוללניים, ועלינו לשקול כיצד בדיוק ניתן יהיה לאמת כל אחד מהם לפני שהפרויקט יעבור לשלב הבא. כל הנושאים האלה נדונים בסעיף 5.4.2.1 של הקווים המנחים, וכל אחת מהסוגיות המזוהות שם מורחבת בתת-סעיף נפרד.

תת-הסעיף של הניהול (5.4.2.2) מאזכר את הגדרת תחומי האחריות, את חלוקת העבודה בתוך הפרויקט, את הממשקים הארגוניים שמחוצה לו, מזהה את לוחות הזמנים לאספקה ואת הדרישה לבקרת התקדמות מתוכננת.

ההיבט היחיד של סעיף 5.4.2.1 שאינו מוסבר בפירוט יתר, הוא הפריט האחרון שנקרא 'ניתוח בעיות פוטנציאליות הקשורות לשלבי הפיתוח ולעמידה בדרישות המפורטות', שלמעשה אינו אלא התייחסות לצורך בניתוח הסיכונים. סיכוני הפרויקטים מטופלים בתוכנית האיכות, שיכולה להיות חלק מתוכנית הפיתוח, ואילו תכנון האיכות הוא נושא סעיף 5.5.

תקן ISO 9000-3 מספק בסעיף 5.5.2, מספר הנחיות שימושיות בדבר תוכן תוכנית האיכות.

ניתוח הדרישות וסקר החוזה

ניתוח הדרישות וסקר החוזה מטופלים בצורה ישירה, ובכלל זה הצעה לתחום האחריות של הנהלת הקונה (כלומר הלקוח). ההתייחסות ללקוחות יכולה להיות רק בלתי ישירה, על ידי כך שהספק מביא לתשומת לב הלקוחות את ההמלצות האלו, משום שהקווים המנחים מיושמים לגבי הספק, ואין הם יכולים להטיל אחריות על הקונה. עם זאת, הקווים המנחים מציינים לפחות את הנושא, ונותנים לספקים את ההזדמנות להצביע לפני הקונים על כך שהקווים המנחים מציעים הנחייה בלתי תלויה לדרכי פעולה טובות.

סעיפים 4.1.2 ו- 4.1.3 מעודדים את הלקוחות לשתף פעולה עם ספקיהם על ידי מסירת כל המידע הדרוש במועדים הנכונים, למנות נציג שיעמוד בקשר עם הספק בנושאים החוזיים, ולהשתתף בסקרים חוזרים משותפים שיתועדו לצורך התייחסות בעתיד. מהקווים המנחים לא ברור מה צריכה להיות תגובת הספק, במקרה שהלקוח אינו מוכן לשתף פעולה. ככלל, תהיה זאת אסטרטגיה לא נכונה להצביע על האיוולת שבדרך התנהגות הלקוח, הדבר יכול להיות גם מסוכן כאשר הלקוח הוא גדול ועשיר. עם זאת, חשוב שיהיה תיעוד כלשהו שייעוץ זה הובא לתשומת לב הלקוח, ושנעשה ניסיון ליישם זאת, לדוגמה, על ידי הקמת מנגנון משותף לסקרים חוזרים.

סעיף 4.1.2 מעלה את השאלה הרגישה של שכנוע הלקוח לקחת חלק באחריות על המוצר שבפיתות. זהו תחום שההתייחסות אליו היא לעיתים תכופות כאל פרט מעשי שרק מעורר רוגז. בדרך כלל, גישת המפתחים היא שהקונה לא יאהב בכל מקרה את מה שמייצרים עבורו; אך גם הקונים פועלים לרוב לפי העיקרון שהמוצר המוגמר לא יענה על צורכיהם. במידה שקיימת איזו שהיא תחושת בעלות על המוצר, תהיה זאת לעיתים תכופות יותר מצד הספקים מאשר מצד הקונים. לכן, לעיתים תכופות הספקים נמצאים בעמדת התגוננות בדבר ביצועי המוצר, ושבעטייה הקונים אינם

דואגים לפתח תרבות עבודה בה יוכל המוצר המוגמר לתפקד במלוא הפוטנציאל שלו. זה מצב המשווע לתחושת שותפות ולהבנה מציאותית של מה שניתן לעשות.

הדגש העיקרי בסעיף 4.1.2 הוא על הצורך לוודא שהקונה ימנה אדם בעל סמכויות מתאימות, שיסכם עם הספק את כל הפרמטרים העיקריים. נציג זה צריך להשגיח גם על הדרישות הטכניות וגם על התנאים החוזיים. חשוב שנציג הקונה ישמש כמקשר יחיד עם הספק, כך שאפשר יהיה לצמצם עד למינימום את העלול להיראות כאי-הבנות שהן בלתי נמנעות. חשוב גם שנציג זה יוכל לאשר אישית מוצרים המתקבלים מהספק, כך שנקודת קשר יחידה זו תיראה על ידי שני הצדדים כמתאימה להשלמתו המוצלחת של החוזה.

כדי לוודא שמנגנון הקישור לא רק יוגדר, אלא גם ייושם בפועל, מציע סעיף 4.1.3 סדרת סקרים משותפים המכסים את השלבים הראשיים של תהליך הפיתוח. למרות שרמה זו של תשומת לב לממשק שבין הספק ללקוח היא עדיין פחות מאידיאלית, היא יכולה להוות בסיס להסכמה פורמלית חשופה פחות לאי-הבנות שבהמשך שעלולות להוליך גם לאכזבות. בסופו של דבר, אין תחליף ליצירת תרבות נכונה המשותפת לשני הצדדים. תחושה של מסע משותף לקראת תוצאה מוצלחת, רבים סיכוייה להפיק מוצרים שישביעו את רצון הקונה, וימנעו מרירות המחלחלת בפרויקטים בהם כל צד מחופר בעמדותיו החוזיות.

תקן ISO 9001 מכיל דרישה לסקר חוזר של החוזה, כאמצעי שנועד לוודא שהקונה והספק יפתחו במשותף הבנה של דרישות הקונה וכוונות הספק למילוי דרישות אלו.

סעיף 5.2 של הקווים המנחים חוזר ומאשר את הדגש שהתקן שם על הצורך בסקר חוזר רשמית של החוזה, ומוסיף מספר פרטים (5.2.1) בדבר האלמנטים שהחוזה האופייני צריך לכסות. חשוב לציין כאן שלא די שהחוזה יעסוק בנושאים מובנים מאליהם, כגון קריטריונים למבחני הקבלה. עליו גם לזהות את השיטה שיש לנהוג לצורך טיפול בבעיות ובשינויים הקשורים בחוזה עצמו. במקום לעסוק בבעיות לאחר שהן מתעוררות, ותוך כדי כך לסכן את מערכת היחסים שבין הקונה לספק. אנו ממליצים לחזות מראש את תחומי הבעיות הפוטנציאליות, ולדון בהן לפני האירוע, באווירה שקטה ובלתי מאיימת.

לבסוף (5.3), הקווים המנחים מגישים ייעוץ בדבר הצגת מפרטי דרישות הקונה. יחד עם זיהוי התוכן המתאים של מפרטי הדרישות, מדגיש סעיף זה את הצורך בשיתוף פעולה הדדי, כך ניתן לוודא שהגדרת הדרישות תיעשה בצורה של פעילות משותפת, בה ייעשה כל מאמץ ליצירת האווירה הנכונה לשיתוף פעולה לטווח ארוך, לכל משך קיומו של הפרויקט.

בקרת הדרישות

לאחר קביעת הדרישות וקבלת הסכמת הלקוח לגביהן, צריך הספק לקיים בקרה על דרישות אלו. כך יוכל לוודא שלא ייעשו בהן שינויים בלתי מורשים או מקריים, שישפיעו על ההסכמה החוזית, ואולי אף יעמידו בסכנה את כל פרויקט הפיתוח.

תקן ISO 9000-3 מכיל שני סעיפים ראשיים העוסקים בבקרת הדרישות: בקרת מסמכים (6.2) וניהול התצורה (6.1). הראשון בא לוודא קיום בקרה על שינויים במסמך הדרישות, ואילו השני, קובע זיהוי ברור של הגירסה ההתחלתית של הדרישות, ומנגנון שיאפשר את המעקב. תוכנה שתיוצר על פי מפרט של דרישות, תזוהה בסופו של דבר כתוצאה של גירסה מסוימת של הדרישות.

כיצד נוודא התחלה מוצלחת של פרויקטים

ניתוח ותיעוד הדרישות

מקובל על הכל שניתוח הדרישות הוא השלב במחזור החיים של פרויקט, שהוא הקשה והחשוף ביותר לשגיאות, והוא גם השלב החשוב ביותר. תיקון טעויות והשמטות בתיעוד הדרישות, יהיה יקר יותר משעה שמתחילים בעבודת הפיתוח, **והדבר עלול גם להוליך לכישלון מלא של הפרויקט**. לכן, נהוג להפקיד את ניתוח הדרישות בידי העובדים המנוסים והמיומנים ביותר שבסגל הפרויקט, וגם אז משתלם לתת כל סיוע אפשרי לביצוע משימה זו.

קיימות כאן שתי בעיות: כיצד מפיקים את הדרישות מהלקוח, וכיצד מתעדים אותן באופן שהתיעוד יהיה נגיש למשתמשים הצריכים לבדוק ולאשר אותו, והמפתחים לא יגלו אי בהירויות בניסוח הדרישות.

אפשר להקל במידה רבה על הפקת הדרישות על ידי טכניקות כגון יצירת אבטיפוס והדמיות. יצירת אבטיפוס של הדרישות משתמשת במודלים ייצוגיים פשוטים של המערכת המוצעת. מטרתם לספק למשתמשים ייצוג מציאותי יתלת ממדיי של הדרישות, בו יוכלו להסתייע כדי לבחון את הדרך בה הם מבינים את השפעתה הצפויה של המערכת שתסופק להם. השימוש באבטיפוס שכיח יותר בתחום הממשק למשתמשים, ופחות במידול פונקציות של המערכת בקנה מידה רחב יותר.

ההדמייה יכולה לסייע בהערכה של הפונקציונליות הכוללת, על ידי מידול תרומתה הצפויה של מערכת התוכנה להתנהגות הכוללת של המערכת הרחבה יותר (כולל תרומתה לבני האדם). בדרך כלל, משתמשת ההדמייה במסמכים מוכנים המחליפים פעולה של מערכת תוכנה במסגרת תרחישים מצומצמים. על ידי תכנון והכנה מדוקדקים אפשר להשתמש בצירוף של אבטיפוס והדמיות, להצגת מודל מקיף של התנהגות המערכת.

אמצעי פשוט שאפשר להסתייע בו ליצירת מסמכים שלמים ועקביים של הדרישות הוא אספקת **מיתאר** (outline) ערוך בצורת **תבנית** (template). התבנית מגדירה את מיתאר מסמך הדרישות ומנחה את מכין המסמך על התוכן הנדרש. המסמכים המתקבלים עקביים יותר, גם בצורתם וגם בתוכנם, כך הם גם קלים יותר לסקירה. השימוש בתבניות המלוות ברשימות תיוג, מסייע גם הוא לאלה המתכוננים לסקר של המסמכים. על ידי בקרה מתמדת של התבניות ורשימות התיוג, ניתן לצבור ניסיון שישפר את תהליך ניתוח הדרישות ויצמצם את המאמץ ועלות הפקת התיעוד הנדרש.

קטלוג הדרישות

קטלוג הדרישות איננו אלא בסיס נתונים שנבנה ממשפטי הדרישות, שהוסרו מהן כל המילים הבלתי חיוניות. בסיס נתונים ממין זה מורכב מאוסף של משפטים פשוטים, שכל אחד מהם מכיל דרישה אחת - טבלה בת עמודה אחת. אנו יכולים לסקור את בסיס הנתונים הזה, ולהחליט אם הוא שלם ומינימלי, ולספק את קו הבסיס של הדרישות לצורך מעקב אחר התיכון ההולך ומתפתח. לאחר הסקירה והאישור, הופך קטלוג הדרישות, בסיס לכל עבודת הפיתוח. תרשים 3.3 מציג את הדרך בה ניתן לבנות את הקטלוג.

מספר דרישה	תיאור הדרישה	קריטריון לקבלה	סעיף מבחן קבלה
1.1	המערכת תספק אמצעי לקליטת הזמנות	בדיקה פונקציונלית	4.1
1.2	אמצעי לסקירת החוזה יהיה זמין ממסך קבלת ההזמנות	בדיקה פונקציונלית	4.2
1.3	אישור ההזמנה יבוקר על ידי סיסמת המשתמש	בדיקה פונקציונלית	4.3
1.4	המערכת תיצור תעודות משלוח אוטומטית עם אישור ההזמנה	בדיקה פונקציונלית	4.4
1.5	המערכת תאפשר קליטת הזמנות באמצעות עד 6 מסכים לפי הצורך, כדי להגיע לקצב קליטה מזערי של 600 הזמנות ליום ו- 1200 שורות הזמנה ליום	בדיקה פונקציונלית 6 מסכים פעילים קצב הזמנות, 600 הזמנות ו- 1200 שורות הזמנה ליום	4.5 4.5.1 4.5.2 4.5.3

תרשים 3.3 *קטלוג דרישות*

עם התקדמות הפיתוח, אנו מוסיפים עמודה חדשה לטבלת הדרישות עבור כל שלב בתהליך. בעמודה החדשה אנו רושמים את האלמנטים של תיכון השלב הנוכחי, המצטרפים למפה של כל אלמנט מהשלב הקודם. בצורה זו, בשלב התיכון הלוגי בו נוצר המפרט הפונקציונלי, אנו מקשרים כל פונקציה אל הדרישה שיצרה אותה. על ידי כך אפשר לאמת כל שלב מול השלב הקודם בכל הנוגע לשלמות, ולוודא שלא תתווסף פעילות כלשהי ואף מוצר שאין עבורם דרישה שהוסכם עליה רשמית. בסוף תהליך הפיתות יש לרשותנו רישום מלא של יכולת מעקב מהדרישות אל המוצרים המוגמרים. בכל שלב אנו גם יכולים להפיק הוכחות לכך שכל אלמנט נוצר מתוך הדרישות, וכי כל הדרישות יושמו במלואן.

קווי פעילות וטבלאות של קווי פעילות

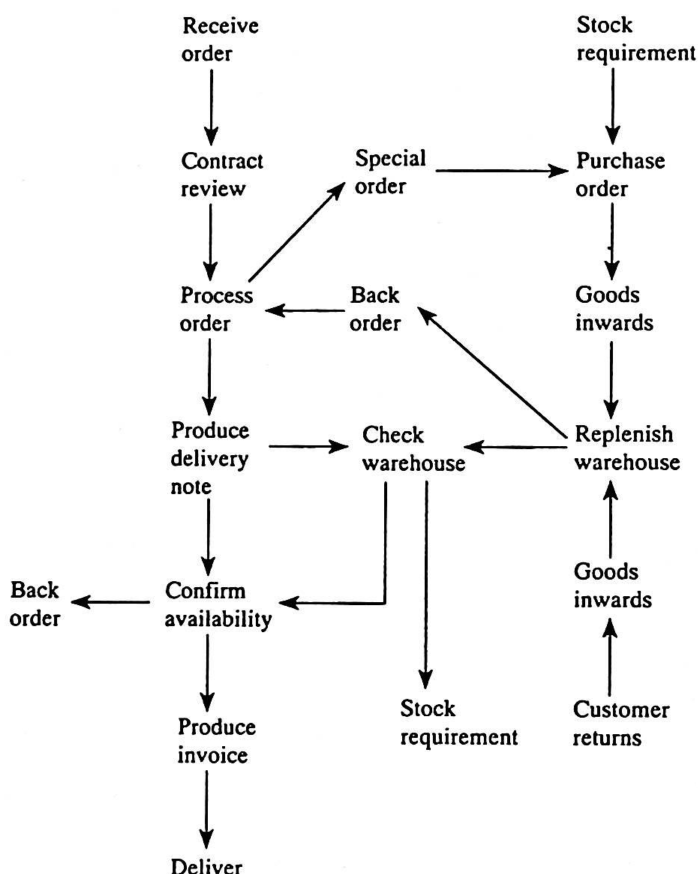
טבלאות של קווי פעילות (activity threads) הן הרחבה של קטלוגי הדרישות, והשימוש בהן קשור לבעיה הכאובה של בדיקות קבלה. בבדיקות אלה יש צורך ברמה כזו של הבנת הדרישות, שתאפשר לקהילת המשתמשים להגדיר מדדי קבלה מכומתים שניתן יהיה לבדוק מולם את המערכת. הגדרת המדדים לקבלה היא אחד ההיבטים הקשים יותר של ניתוח הדרישות, ורק לעיתים נדירות היא מתבצעת כהלכה. בדרך כלל, הנימוק המושמע הוא שהדרישות אינן מובנות די הצורך עד כדי יכולת להגדיר מדדי קבלה. במקרה והדבר נכון, הרי שהדרישות מייצגות בסיס רעוע מכדי שאפשר יהיה לבסס עליו את הפיתוח. למעשה, יכולתנו לבנות מדדי קבלה מכומתים, היא אחד המבחנים הטובים יותר לאיכות קבוצת הדרישות.

כיצד יכולה טבלת קווי הפעילות לסייע לנו? מכיון שטבלה זו היא גם קטלוג של דרישות, הרי שבמקרה והכל תקין, היא אמורה להכיל קבוצה של דרישות בדידות. עלינו לעבור עכשיו לאחור משלב הדרישות, כדי לאחזר את הפעילויות העסקיות מהן הופקו הדרישות. הפעולות העסקיות, קווי הפעילות, הן הבסיס הטוב ביותר להבנת המשתמש את ההשפעה הפוטנציאלית של המערכת שבפיתוח. פעולות אלו תהיינה גם התיאור הנגיש ביותר של התנהגות המערכת, שעל פיו ניתן יהיה להגדיר מדדים להצלחת הפרויקט. בניית טבלת קווי הפעילות היא חלק חשוב בתהליך הקמת בסיס יציב לפיתוח.

תרשים 3.4 מציג כיצד ניתן לשלוף את קווי הפעילות מתוך ידיעת התהליכים העסקיים הבסיסיים. התרשים הוא מודל פשוט של התהליכים העסקיים הבסיסיים של חברה למכירת ספרים בדיוור ישיר. בצד שמאל, ההזמנות מתקבלות ונבדקות לפני העיבוד והעברתן למחסן. המחסן בודק ומאשר את זמינות הסחורה, לבסוף מכינים את החשבונית. רצף זה מייצג את קו הפעילות הפשוט ביותר, בו מתקבלות הזמנות שגרתיות עבור ספרים רגילים הנמצאים במלאי. כל המצבים האחרים של ההזמנות, כגון מלאי שאינו מספיק, הם וריאציות על הנושא הבסיסי. בצד ימין אנו רואים את התהליך הבסיסי של השלמת המלאי, ובאמצע מופיעות אחדות מהפעולות המתבצעות במחסן. ברור שאחדים מהתהליכים המוצגים יהיו ידניים ואחרים ממוחשבים, וחלק מהם יניעו בוודאי תהליכים במערכות אחרות.

בזמן ניתוח הדרישות, תפקיד המנתח ליצור קבוצה עקבית וסבירה של דרישות מוגדרות היטב. המנתח מנסה להמיר את תיאור דרישות המשתמשים, תיאור המבוטא בשפה שלהם וספוג בתרבותם, למשפטי דרישות מנוסחים בבהירות, בתמציתיות, ניתנים לכימות, וכיוצא באלה. לתהליך זה יכולות להיות תופעות לוואי לא רצויות יכול להיות שמשפטי הדרישות יהיו מובנים פחות למשתמשים מאשר הגירסה המקורית של דרישותיהם. כתוצאה מכך, ייתכן שההסכמה הרשמית לדרישות שלאחר הניתוח, לא תיתן למשתמשים את מידת הביטחון שהיינו רוצים שתהיה להם. הם אפילו עשויים להתקשות בהמרת דרישותיהם, כפי שהם מבינים אותן, לקבוצה ברורה של קריטריונים לקבלת המוצר.

מסתבר שזה המחיר שעלינו לשלם עבור הניסיון להגיע לדרישות מוגדרות היטב. עם זאת, נוכל לנטרל מגבלה זו אם נמשיך להחזיק במשפטים המקוריים של הלקוח, המתייחסים לכוונות הכלליות, שאולי אינם מוגדרות די הצורך. משפטים אלה, לאחר שיפוץ מסוים שמטרתו הבהרת הביטויים, יוכלו לשמש בתור עמודה ראשונה של טבלת קווי הפעילויות, שתבוא לפני הדרישות עצמן.



***תרשים 3.4* ניתוח קו פעילות עבור תהליך הזמנה בדואר**

כל קו בטבלת קווי הפעילויות עשוי להביא ליותר מאשר דרישה אחת. אנו יכולים להשתמש בטבלת הקווים כדי שתסייע לנו בסקירת הדרישות, כדי לוודא שכל הקווים מוצאים את ביטויים בדרישות, וכי הדרישות מבטאות רק את כוונות המשתמשים כפי שהובעו בקווי הפעילות. תרשים 3.5 מציג את הדרך בה אפשר לבנות את לוח קווי הפעילויות.

קו 1 הוא תהליך ההזמנות הבסיסי, והוא משמש כקו הראשון משום שזהו התהליך היסודי ביותר. אם תהליך זה לא יפעל בצורה נכונה, אין סיכוי שאף אחד מהתהליכים יוכל לפעול (המספרים מתיחסים לפירוט בתרשים 3.5 שלהלן). קו 1 מחולל שש דרישות של המערכת, עבור הצעדים השונים שבתהליך. לצעדים 1.1 ו- 1.2 יש רכיב

משלים ידני שצריך להתבצע על ידי פקידי ההזמנות; דבר זה צריך להשתקף בנהלים המתאימים. מצעד 1.5 משתמע חיבור אל מערכת ניהול המחסן, שהיא מערכת מחשב נפרדת. כך שבשלב כלשהו חייבים לבוא הגדרה ובדיקה של שילוב מערכות. בעמודה הבאה, רוב הדרישות מחוללות פונקציית מערכת יחידה. עם זאת, דרישה 1.2 זקוקה להגדרה מפורטת יותר, ותיושם על ידי שתי פונקציות נפרדות. קו 2 מגדיר את הקו השני בחשיבותו, חידוש פשוט של המלאי, וכן הלאה.

קו	תיאור הקו	דרישות המערכת	פונקציות
1	הזמנות רגילות (ללא הזמנות ממתינות וללא 'מיוחדים')	1.1 קליטת הזמנות	1.1
		1.2 סקירת החוזה	1.2.1 חיפוש בבסיס נתונים
			1.2.3 השהיית הזמנה
		1.3 אישור ההזמנה	1.3
		1.4 יצירת תעודת משלוח	1.4
		1.5 אישור זמינות	1.5
		1.6 הכנת חשבונית	1.6
2	חידוש מלאי (ללא מיוחדים)	2.1 בדיקת המלאי	2.1
		2.2 הפקת פקודת רכש	2.2
		2.3 קבלת סחורה	2.3
		2.4 חידוש מלאי	2.4.1 עדכן רמות מלאי
			2.4.2 עדכן הזמנות ממתינות
			2.4.3 הוסף החזרות מלקוחות
3	ניהול המחסן	3.1 קבל תעודת משלוח	3.1
		3.2 אשר זמינות	3.2
		3.3 בדיקת סחורה נכנסת	3.3.1 בדוק מול פקודת רכש
			3.3.2 בדוק מיוחדים
		3.4 החזרות מלקוחות	3.4.1 בדוק מצב
			3.4.2 הוסף למלאי
		3.5 עדכן רשומות מלאי	

***תרשים 3.5** טבלת קווי פעילויות של חברה למכירת ספרים בדיוור ישיר*

טבלת הקווים שבתרשים 3.5 אינה שלמה, משום שהיא מציגה רק אחדים מהקווים וכוללת ניתוח רק עד לרמת הפונקציה. ניתן להמשיך את התהליך דרך תהליך הפיתוח, תוך הוספת עמודות לטבלה בכל פאזה, עד שכל מודולי הקוד וכל היישויות של בסיסי הנתונים יהיו מקושרים אל קו אחד או אל מספר קווים. צוות הפיתוח יהיה מסוגל לוודא בכל שלב, שהתיכון תואם בדיוק את תיאור השלב הקודם, וכך ייושמו כל הדרישות בשלמותן, ולא ייושם דבר שלא צוין כדרישה.

כעת מכילה טבלה מוגדלת זו לא רק את דרישות הבסיס, אלא גם את קווי הפעילות העסקית ממנה הן נובעות. כך אנו מקיימים את המגע בין המשתמשים לבין המערכת המתפתחת; הם יכולים לקשר כל דבר חזרה לאחור אל רעיונותיהם שהובעו בדרכם. למשתמשים קל יותר יהיה לכמת ולתת עדיפויות לקווים מאשר לדרישות, וגם קל יותר יהיה להגדיר קריטריונים לקבלה, ולתכנן את בדיקת הקבלה.

בשלב הפיתוח המוצג בתרשים 3.5, המשתמשים אמורים להגדיר כבר את גורמי ההצלחה הכלליים של הפרויקט, ואת המדדים לקבלה לגבי כל דרישה. מצביעי הצלחה קריטיים הן המדידות שבעזרתן יחליטו המשתמשים אם הושגו יעדיהם העסקיים. אלה מוגדרים בדרך כלל ברמה של הקו, כלומר, קצב עיבוד ההזמנות. המדדים לקבלה הם אלה שיש עליהם הסכמה בין המשתמשים והמפתחים כעל מדדי הקבילות של מערכת התוכנה בעת הקבלה, כלומר, זמנים ממוצעים לחיפוש בבסיס הנתונים. מהמדדים של המשתמשים אמורים המפתחים להגדיר גורמי מפתח של הביצועים, שישמשו כמדדים טכניים לביצועי המערכת, שנדרשים לעמידה במדדים של המשתמשים. לדוגמה, מהירות מינימום של המעבד תיקבע לפי פרמטרים כגון אלה הנדרשים בהתאם לקצב העיבודים, ולגודל הצפוי של בסיס הנתונים. תרשים 3.6 מדגים את הקשרים שבין מדידות אלו.

טבלת הקווים היא גם מכשיר רב ערך להגדרת אסטרטגיית הבדיקות. צריך יהיה לדרג בקפדנות את סדר העדיפות של כל הבדיקות המובנות המצריכות תכנון, החל ממודולים ייחודיים של תוכניות ועד מערכות שלמות. כך עם הופעת לחצי הזמן הבלתי נמנעים, נוכל להחליט החלטות שקולות בדבר ביטול, או שינוי תוכנית בדיקה כלשהי. דירוג סדר העדיפות של הבדיקות מאפשר להעריך חשיבות של קבוצת בדיקות נתונה, ואת המשמעויות הצפויות כתוצאה מקיצוץ, או ביטול בדיקות אלה. על ידי כך, לא רק נימנע מקיצוצים סיטוניים ומנוגדים-לפריון בתוכנית הבדיקות, אלא נהיה גם בעמדה שתאפשר לנו לשקול את הסיכונים הנלווים לכל קיצוץ שנחליט עליו. על ידי הגדרת מדידות מתאימות עבור פעילות הבדיקות, נוכל לכמת את השלמות ויעילות הבדיקות, וכך לקבוע רמות סיכון למקרה שהבדיקות יקוצצו באחד השלבים. פיתוח אסטרטגיית בדיקות המבוססת על טבלת הקווים תידון ביתר פירוט בפרק 4.

היתרון הסופי שאנו מפיקים מטבלות הקווים, **הוא היכולת להציג את הציפיות בהמשך הדרך**. הקווים רושמים את הציפיות שמעבר לקבלת המוצר, וגם בתוך פרק השימוש התפעולי של המערכת. שישה חודשים ומעלה לאחר קבלת המערכת, נוכל להעריך אם הפקנו ממנה את היתרונות המתוכננים.

ניתוח דרישות וניהול פרויקט מתוך ההֶקְשר

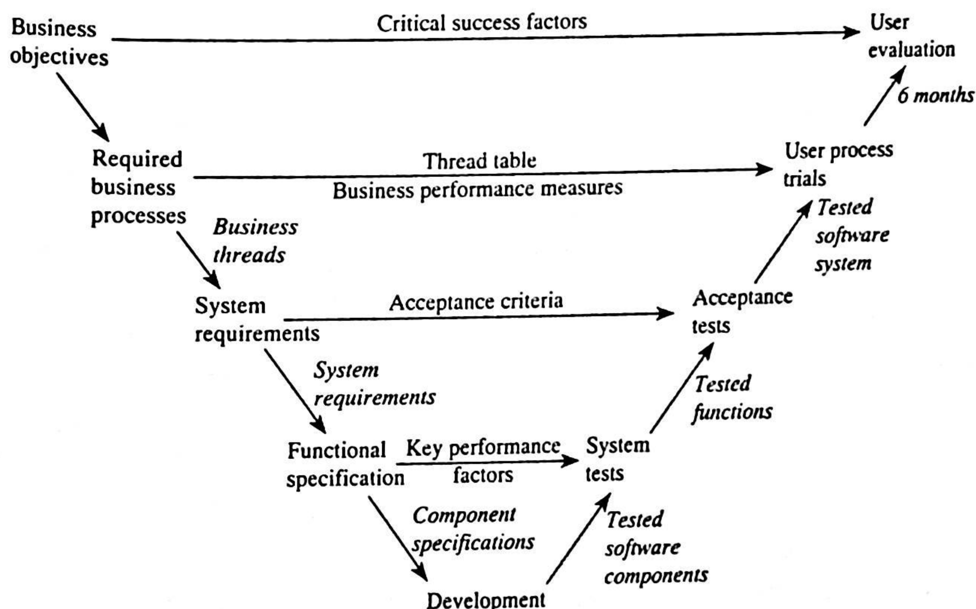
בעולם אידיאלי, לקוחותינו היו מחליטים מראש מה נדרש להם ומעבירים לנו את דרישותיהם. כך היינו יכולים לנתחן ולבנות מערכת תוכנה שתענה על דרישות אלו. את הסיום המוצלח של הפרויקט שלנו היינו מוודאים על ידי ביצוע כל העבודה, לפי תוכנית שהיתה מותווית בתחילת הפרויקט ומבוססת על המתודולוגיה שבחרנו כמתאימה ביותר עבור הפרויקט.

בפועל, החיים אינם פשוטים כמו האידיאל. ככלל, הלקוחות אינם מסוגלים להגדיר את דרישותיהם במלואן, בבהירות ובצורה חד-משמעית; התקשורת רחוקה מלהיות מושלמת ואף מוסיפה שגיאות ואי-רלוונטיות שמסיטות אותנו מהסוגיות החשובות יותר, בנוסף איננו בטוחים לגמרי בשיטות הפיתוח שלנו. כל הפגמים האלה מצטברים ומהווים מקור דאגה ממשי למנהלי פרויקטים - **הסיכון**.

הסיכון הוא ההסתברות המוחלטת שהעניינים אכן ישתבשו ויגרמו לפרויקטים שאנו מפתחים לסטות מהתוכנית. אם לא מטפלים בהם במהירות ובהחלטיות, הסיכונים יובילו לבעיות כבדות, ואולי אף לכישלון מוחלט. בשל תופעת הסיכון, אנו חייבים לתכנן בזהירות רבה יותר, ולכלול בתוכניתנו את הפעולות שעלינו לנקוט כדי לנטרל את הסיכונים שביכולתנו לחזות. הערכת סיכונים היא חלק חיוני של התכנון, וניהול סיכונים הוא חלק יסודי של ניהול פרויקטים.

בהתחשב בהשפעה השלילית שיש לסיכונים על תוכנית העבודה שלנו, עלינו להיות בטוחים שאנו מבינים את מה שהפרויקט שלנו אמור להשיג ושהגדרנו את התהליך (או את המתודולוגיה) בפירוט. כך נוכל לפקח על ההתקדמות ולגלות כל חריגה מהתוכנית כבר בשלב מוקדם, כאשר האחזור אפשרי עדיין, וללא השלכות חמורות על העלויות, או על לוחות הזמנים.

הגדרת **מודל מחזור חיים** מספקת מסגרת לתכנון פרויקטים מציאותי, ומהווה את הבסיס למתודולוגיה כוללת לניהול פרויקטים. על ידי ניתוח ותיעוד הדרישות בפירוט רב ככל האפשר, ניתן להגדיר נקודות התחלה וסיום ברורות של הפרויקט. אנו משליכים מהדרישות קדימה, לעבר סיום הפרויקט, על ידי הגדרת מדדי הקבלה, המבוססים אך ורק על הצגת הדרישות שבקטלוג הדרישות. טכניקות כגון טבלת קווי הפעילויות, מסייעות לוודא שהלקוחות יישארו בתמונה של המוצרים המתפתחים, ויוכלו לקשרם אל היעדים העסקיים מהם נבעו הדרישות. גישה זו מעודדת את הלקוחות לשמר את תחושת הבעלות על המוצרים במשך הפיתוח, ומעודדת את הדו-שיח בין הלקוחות לבין המפתחים.



תרשים 3.6 מדידת ביצועי המערכת

מדידות שמקורן ביעדים ההתחלתיים של הלקוח ושנבנו סביב הצגת הדרישות, יכולות לספק לנו אבני דרך במפת הדרכים לאורך תהליך הפיתוח, ומאפשרות לנו להיות בטוחים שהגענו אל הנקודה הנכונה, כאשר הפרויקט מסתיים. תרשים 3.6 מציג את מנגנון הבקרה של הלולאה הסגורה שנוצר על ידי 'מדדי ההצלחה' אלה.

כעת, כשעומדת לרשותנו המסגרת לתהליך ניהול פרויקטים, נוכל להפנות את תשומת הלב להגדרת תהליך הפיתוח בפירוט רב יותר.

עקרונות בסיסיים של ניהול פרויקטים

בפרק 2 זיהינו אחדים מעקרונות איכות התוכנה וסיכמנו, שמערכת איכות מציאותית חייבת לענות על עקרונות בסיסיים אלה. דבר זה נכון במידה שווה גם לגבי ניהול פרויקטים.

ניהול פרויקטים הוא נושא רחב ומורכב, וטיפול מקיף בנושא זה היה חורג מתחום ספר זה. לעומת זאת, ניהול מציאותי של פרויקטים אינו יותר מאשר היצמדות למספר מועט של עקרונות חשובים. נציג כאן מספר עקרונות בסיסיים שיצביעו על סוגי הנושאים בהם יש צורך לטפל. אין לצפות שרשימתנו תהיה שלמה, ואפשר יהיה גם לטעון שיש נושאים חשובים יותר מאלה שנכללו בה. הנקודה החשובה היא רכישת מודָעות לקיום העקרונות.

עשרת העקרונות של ניהול פרויקטים

---

1. **פיתוח תוכנה הוא תהליך הכולל חזרות (איטרטיבי), וכך גם צריך לתכננו.**

המשימות תתבצענה טוב יותר בשניים או בשלושה שלבים. תכנן לקראת אספקה מהירה של טיוטות, עבור עליהן ועבד אותן מחדש.

2. **תכנן עריכת בדיקות.**

הקצה בתוכניותיך זמן מספיק להשגת רמה טובה של אימות ובדיקת תקפות. בדרך כלל יוקדשו כ- 40 אחוז מהמאמץ הכולל לפעילות זו.

3. **אם אינך מסוגל לכלול בלוח הזמנים גם את הפיתוח וגם את הבדיקות והניסויים, צמצם את דרישותיך או ספק אותן בשלבים.**

מוטב לאכזב במקצת את המשתמשים ואחר כך לספק את המוצר במועד, מאשר לגרום להם לרוגז רב בשל אספקה מאוחרת מבלי לענות על הדרישות.

4. **ערוך בדיקות לפי תוכנית.**

לאחר שהוקצה זמן לבדיקות, תכנן זמן זה בקפדנות כדי שתוכל להפיק מהבדיקות את הערך המירבי בזמן ביצוען.

5. **היה מציאותי.**

אם העבודה אינה יכולה להתבצע בזמן שהוקצה, צמצום הערכות הזמן לא יביא כל תועלת. יש להניח שהיית יותר מדי אופטימי כבר מלכתחילה.

6. **הכן תכנון מפורט.**

פצל את הפעילויות לרמה שתתן לך אפשרות לראות את קצב ההתקדמות על בסיס שבועי.

7. **הקפד על קיום בקרה.**

בדוק את ההתקדמות בצורה סדירה ובמקרה של פיגורים, ברר את הסיבה לכך. תמיד קיימת סיבה, ואם אין מטפלים בה, הפיגורים ילכו ויגדלו.

8. **פקח עין על השינויים.**

שינויים בדרישות יכולים להיות יקרים ביותר וחייבים להעריכם בזהירות. כל השינויים האחרים מחייבים אף הם פיקוח. אלה עלולים להכיל גם פצצות זמן.

9. **הכן עצמך לקראת הופעת פעילויות 'כמעט קריטיות'.**

הנתיבים הקריטיים נוהגים להתחלף מדי פעם. כאשר הדבר קורה, הם חושפים פעילויות קריטיות חדשות שאין לך כמעט שליטה עליהן. פקח עין על פעילויות אלו.

10. **שמור על עדכון המדדים.**

החלט מה הם הדברים החייבים מדידה והמשך למדוד אותם בכל תנאי. מידע זה הוא בעל ערך רב ביותר כאשר אתה 'ממש נתקל בו', וזה הזמן שדווקא בו מתעורר הפיתוי לחדול מאיסוף הנתונים.

הכן לעצמך את רשימת עקרונותיך, והשתמש בה כדי לעורר דיון בסוגיות המפתח ובחשיבותן היחסית. לכשתגיע להסכמה כללית, בנה סביב לעקרונות המוסכמים את כללי ההתנהגות שלך לניהול פרויקטים, ודאג שכל התחום הזה יהיה נתון לסיקור מתמיד. משימתנו הראשונה היא גיבוש תהליך מוסכם ושימושי על ידי העלאתו על הכתב. משעה ששיטתנו עומדת לרשותנו בכתב, נוכל לשקול דרכים לשיפורה.

פרק 4

תהליך הפיתוח הבסיסי

מבוא

טיפול נכון בתהליך הפיתוח הבסיסי הוא מפתח לייצור ותחזוקה מוצלחים של תוכנה. הפיתוח על פי מחזור חיים משקף את תרבות הארגון בכך שהוא מגדיר פאזות ומוצרים מוגמרים מציאותיים של מחזור החיים. בניית מודל של מחזור חיים אידיאלי יכולה להיות מענגת למדי, אך לא תהיה בו תועלת רבה, אם לא נוכל להיצמד למבנה המוגדר על ידו. יישום מעשי של תהליך פיתוח מציאותי ומוגדר היטב, יכול לוודא את השגת סביבת האיכות הטובה ביותר עבור כל יחידת עבודה שאנו מקבלים על עצמנו. המאמץ הכרוך ביצירת תהליך פיתוח מתועד אינו זניח, אך היתרונות המופקים - מפצים בהרבה על המאמץ.

מהו התהליך הבסיסי

תהליך פיתוח התוכנה הבסיסי הוא התהליך היחיד החשוב ביותר שעל מערכת ניהול האיכות ללכוד. אם נסלק ממנו את כל הכלים, הטכניקות והשיטות המסייעות לו, או בולמות אותו, תהליך הפיתוח הבסיסי יורכב רק מארבעת האלמנטים החיוניים:

- המסגרת עבור התהליך - מודל של מחזור חיים;
- קבוצה מסוימת של שלבים ומוצרים מוגמרים - מתודולוגיה;
- אמצעים שיאפשרו לוודא אם המוצרים המוגמרים נכונים, אם לאו - אימות ובדיקת תקפות;
- אמצעים לזיהוי מוצרי תוכנה ולבקרת השינויים - ניהול התצורה.

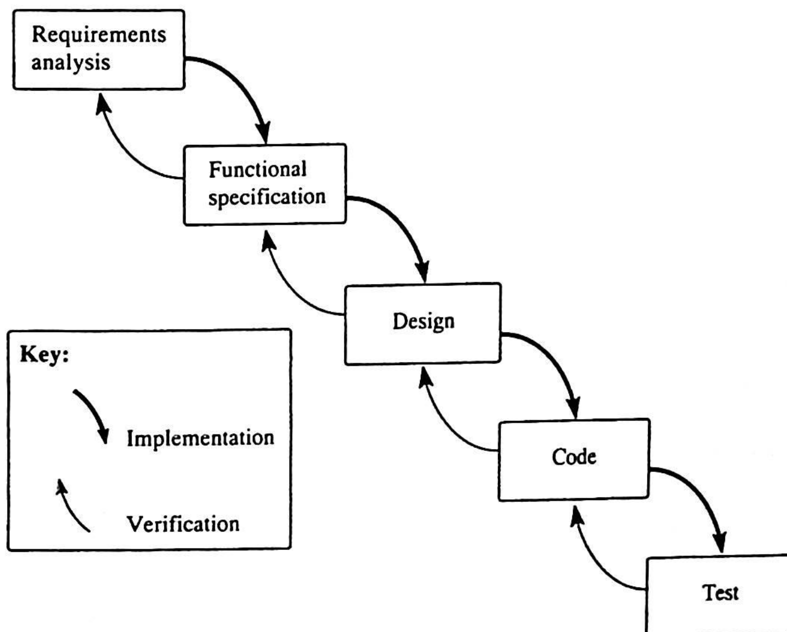
כשארבעת אלמנטים אלה מצורפים יחד, הם תואמים את שמונת העקרונות שנדונו בפרק 1.

קיים עוד תחום חמישי בעל משמעות רבה, שאינו בהכרח חלק מתהליך הפיתוח הבסיסי, ועם זאת, מהווה חלק ראשי של מחזור החיים הכולל של מוצר תוכנה. הכוונה לנושא **תחזוקת התוכנה**. אנו דנים בו בפרק זה בשל הקשר ההדוק שלו לגישת הפיתוח הבסיסית. לכל דבר יש חשיבות רק במידה שהוא תורם לאחד מארבעת האלמנטים האלה, בהם נעמיק כעת את הדיון.

מחזורי חיים

מודל מחזור חיים של פיתוח תוכנה אינו אלא מיפוי של התהליך הכולל, לא יותר ולא פחות מזה. הוא מגדיר מהם השלבים שהתהליך מגלם, וכיצד הם קשורים זה לזה. קיים מגוון רחב של מודלים 'סטנדרטיים' לייצוג מחזור חיים, וכל אחד מהם מציע גישה שונה אל משימת הפיתוח. עם זאת, כל ארגון העוסק בפיתוח תוכנה זקוק למודל, או למודלים שישקפו את התרבות והפילוסופיה שלו. מודלים מעשיים של מחזור חיים יכולים להיות מבוססים על מודלים 'סטנדרטיים', אך אין הכרח בכך. ואכן, רק ארגונים מעטים מיישמים את הגישה ה'סטנדרטית' לפיתוח התוכנה.

מודל מפל המים

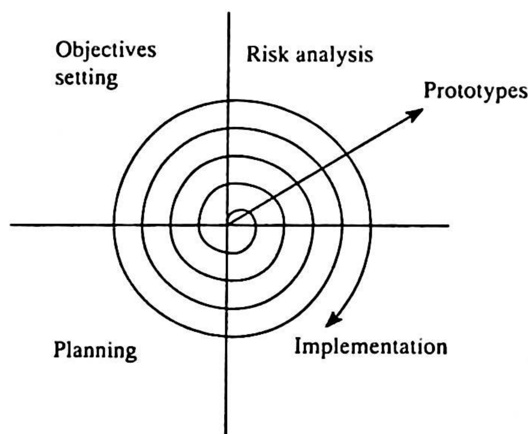


תרשים 4.1 מודל מפל המים

תרשים 4.1 מציג את מה שהוא אולי הווריאציה הפשוטה ביותר של **מודל מפל המים**. במודל זה, התהליך הבסיסי הוא **סודר** (sequential). למרות שייתכנו חזרות בכל אחד מהשלבים, התהליך עצמו מתקדם משלב לשלב בצורה קווית רציפה. הרצף כולל בדרך כלל בניית מפרט דרישות בשיתוף המשתמשים, ואחר, שימוש במפרטי הדרישות להפקה של מפרט פונקציונלי שעל המפתחים ליישם. אחר בא מפרט התיכון שמומר מאוחר יותר לקוד, ובו מתבצעות בדיקות.

מודל זה זכה לביקורת ממספר סיבות, החשובה שבהן היא העובדה שמשאירים את הבדיקות לשלב מאוחר מאוד בתהליך. הרחבות של תהליך זה כוללות פעולות אימות בכל שלב, אך הרצף ממשיך להיות המאפיין הראשי שלו. מודל הפיתוח של מפל המים נמצא בשימוש נפוץ ביותר במפעלי תעשייה, ולעיתים תכופות כברירת מחדל במקומות בהם לא הוגדר מודל פיתוח רשמי אחר.

מודל החילזון



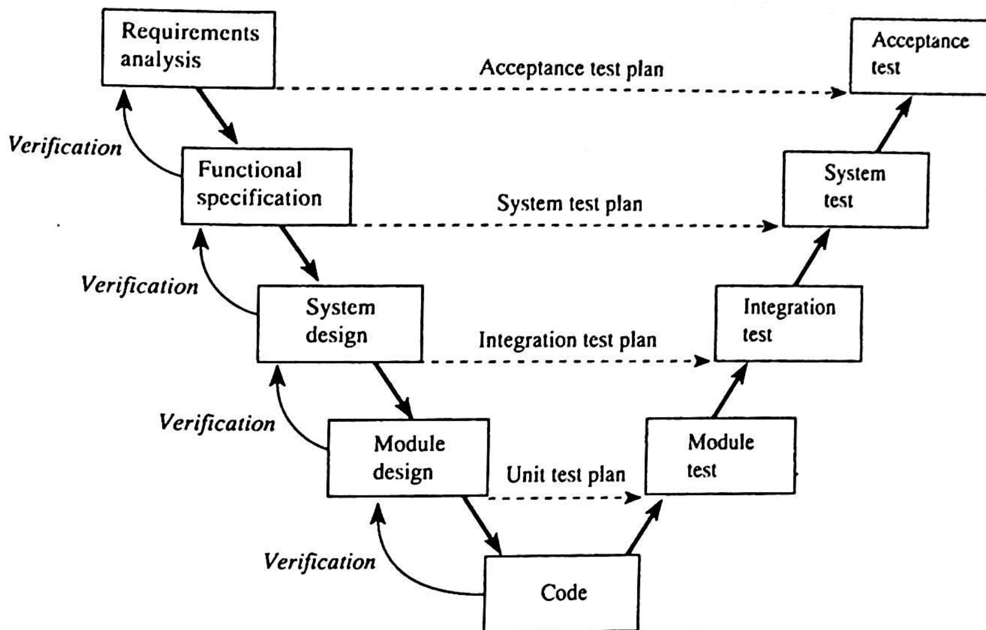
***תרשים 4.2** מודל החילזון*

תרשים 4.2 מציג את **מודל החילזון** (spiral). בניגוד למפל המים, מודל זה הוא בתמציתו **איטרטיבי**. יש צורות רבות של המודל, אך כולן מדגישות את האופי האיטרטיבי של התהליך, אף שאחדות מהווריאציות שלו ממליצות שהשלב האחרון בפיתוח החלזוני יהיה פיתוח של מפל מים באיטרציה הסופית של התיכון.

על פי המודל החלזוני בונים סדרה של אבטיפוסים, שכל אחד מהם מוערך אל מול הדרישות עד שמתגלה פתרון קביל. בנקודה זו אפשר להשתמש באבטיפוס לצורך יצירת מפרט דרישות ואז לסלקו, או שאפשר להציג ולמסור את האבטיפוס, בתור המערכת שעונה על הדרישות.

המודל ממליץ שהגישה אל הפיתוח תהיה ללא **הגדרה סופית** (כלומר פתוח), גישה שקשה יהיה לנהלה באמצעות טכניקות מסורתיות של ניהול פרויקטים. מודל החילזון לא היה עד כה בשימוש נפוץ, אך הוא מתחיל להיות מקובל יותר, ככל שגדלה הפופולריות של טכניקות האבטיפוס. טכניקות חלזוניות, או 'לא-קוויות', יידונו בחלק 4 של הספר.

מודל מסוג V



***תרשים 4.3** מודל מסוג V*

מודל מסוג V מוצג בתור פיתוח של מודל מפל המים, והוא הולך ונעשה פופולרי יותר ויותר. מודל V הוא דוגמה של המודל התיאורטי שרק לעיתים רחוקות מיושם בשלמותו, אך עם זאת, משמש כהשראה לרבים ממחזורי החיים המעשיים.

תרשים 4.3 מציג מודל מסוג V. למרות שבתמציתו זהו מודל המבוסס על רצף, המודל שלפנינו מתרכז בבדיקות בשלב הרבה יותר מוקדם מאשר מפל המים. אופי האימות ובדיקת התקפות שבמודל, והתפקיד והמטרה שיש למתודולוגיה, יידונו בפירוט מסוים בהמשך הפרק. בפרקים הבאים ישמש המודל כאמצעי לתיאור והשוואת שיטות וטכניקות.

מתודולוגיות

מהי **מתודולוגיה**? כיצד נכיר אותה לכשניתקל בה?

> ***מתודולוגיה** - אוסף שיטות שבשימוש בענף פעילות מסוים.*
>
> ***מילון אוקספורד המקוצר***

מכל בחינה מעשית, מתודולוגיה של פיתוח תוכנה היא שיטה, או שיטות אשר ישימות לאחת או יותר משלבי מחזור חיים; בתנאים אידיאליים תהיה המתודולוגיה ישימה לגבי כל השלבים.

המתודולוגיות לפיתוח תוכנה כוללות בדרך כלל:

- ייצוג גרפי ו/או טקסטואלי;
- הגדרה של השלבים שיינקטו ומטרותיהם;
- הגדרה של הקלט והפלט הדרושים עבור כל שלב;
- תבניות של המוצרים המוגמרים הנדרשים מכל שלב.

ההגדרה אינה מכתיבה כללים חד-משמעיים. כל דבר שיענה על המדדים הרשומים לעיל, יהיה בו כדי לספק את רוב היתרונות שניתן להפיק ממתודולוגיה, כל עוד הוא הולם את צרכינו.

לשם מה להשתמש במתודולוגיות? האם אין בהן משום תקורה ביורוקרטית המאיטה את קצב הפיתוח? הדבר ייתכן מאוד! למעשה, בחירה במתודולוגיה שאינה הולמת, כגון מתודולוגיה המכילה יותר שלבים, או דרישות לרישום מכפי הדרוש לנו, תגרום לעיתים תכופות לתקורה מיותרת ותצמצם את הפריון. עם זאת, יש להיזהר מהמסקנה האומרת שכל דבר נחשב לתקורה רק משום שהוא גורם לנו לעשות דברים שקודם לכן לא נהגנו לעשותם.

כדי שמתודולוגיה תוסיף עבורנו ערך, עליה להגביר בדרך כלשהי את הפריון שלנו. כיצד יש ביכולת המתודולוגיה לעשות זאת?

1. על ידי הקטנת מספר השגיאות והעיבודים החוזרים הנלווים להן, בעיקר כאשר מדובר בשגיאות הנעשות בשלבים המוקדמים של התהליך, שבדרך כלל גורמות לעבודה חוזרת יקרה לקראת סופו של מחזור החיים.
2. על ידי צמצום מספר השגיאות שעלינו לתקן לאחר האספקה, דבר המאפשר לנו לצמצם את המשאבים המוקצבים לתחזוקה.
3. על ידי כך שהיא מאפשרת לנו להגדיר כלים שיסייעו בפעילויות היותר שגרתיות, ולקשר יחד את הכלים ליצירת סביבת פיתוח אפקטיבית.

הסיבה הראשית לשימוש במתודולוגיה היא היתרונות הנובעים משימוש באוסף שיטות עקביות וסבירות.

היכולת להגדיר ולנצל שפה משותפת, מילולית וגרפית, מהווה צעד עצום לקראת הסרת מחסומי התקשורת שקיימים באופן בלתי נמנע בין הלקוח, המפתח ושלבי הפיתוח. ללא שפה משותפת זו, לא יהיה בסיס לדבר חשוב יותר, והוא תרבות משותפת. כאשר אנשים מתחילים להבין את המידע המועבר להם, גדלים הסיכויים שיהיו אוהדים יותר לנקודת הראות של מעביר המידע. תועלת יחידה זו כשלעצמה יש בה כדי להצדיק לחלוטין את המאמץ של יישום המתודולוגיה, אלא שעל אלה מתוסף יתרון נכבד.

גם מנהל הפרויקט יוצא נשכר מגישת המתודולוגיה. המנהל נמצא תחת לחץ רב שנגרם ממאמציו להעריך את עלות העבודה ולהרכיב תוכנית סבירה עבור הפיתוח. המתודולוגיה יכולה לסייע לעדן את מיתאר הפרויקט שנוצר על ידי מחזור החיים הנבחר. משעה שמנהל פרויקט מחליט על מחזור החיים הבסיסי, המתודולוגיה, מבנה

השלבים, הסיכונים, ופעילויות האיכות המתאימות שינהלו את הסיכונים, מאותה שעה, למעשה, הפרויקט כבר די מתוכנן ורמת הפירוט תגרום לכך שיהיה קל יותר לאמוד את הפעילויות.

כיצד ניתן ליישם את המתודולוגיה למודל V? פשוט, על ידי הגדרת המוצרים המוגמרים שיתקבלו מכל שלב, כולל צורתם, וקביעת הדרך בה יאומתו. לדוגמה, נוכל להגדיר את פלט שלב ניתוח הדרישות כדלקמן (הקודים מתייחסים למסמכי תבנית הפרויקט של האירוע שבדוגמה בהמשך):

1. **מפרט דרישות**, שצריך להכיל את הכותרות שבתבנית RS1 ולכסות את הנושאים שברשימת התיוג CK1. את ההקשר הכולל יש לתאר בתרשים זרימת נתונים הנתמך על ידי מילון נתונים ומפרטי תהליך מתאימים. כללים לבניית תרשימי זרימת נתונים כלולים ב-COP1.
2. **מפרט של מבחן קבלה**, שחייב להכיל הפנייה צולבת אל כל דרישה נפרדת ולהצביע על הקריטריונים לקבלת כל דרישה. מפרט בדיקת הקבלה צריך להיות ערוך על פי תבנית AT1 ולענות על הסוגיות שהוגדרו ב-CK2. יש לערוך טבלה שתראה את הקשרים בין כל הדרישות לבין מפרטי בדיקתן.
3. **אימות שני המסמכים** צריך להיעשות תוך בדיקה חוזרת של התיכון בנוכחות הרשויות הקובעות את הדרישות, את התיכון, ואת הבדיקות ולפחות נציג אחד של המשתמשים.

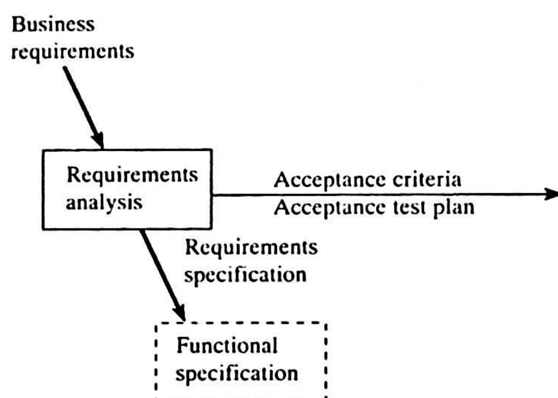
אם כל שלבי מחזור החיים של הפיתוח יוגדרו בדרך זאת, תתקבל תוצאה שתהיה מתודולוגיה שימושית.

אימות ובדיקת תקפות

שני המונחים האלה נדונים לרוב כאילו הם מהווים חלק של טכניקה יחידה. דבר זה נכון מבחינות מסוימות. אף מוצר אינו יכול להיחשב כראוי לשימוש, אלא אם כן עבר **אימות ובדיקת תקפות** (verification and validation), אלא ששתי הפעילויות נבדלות זו מזו ומשיגות מטרות שונות. נטיב להמחיש זאת על ידי עיון חוזר בחלקים המרכיבים את מחזור החיים V.

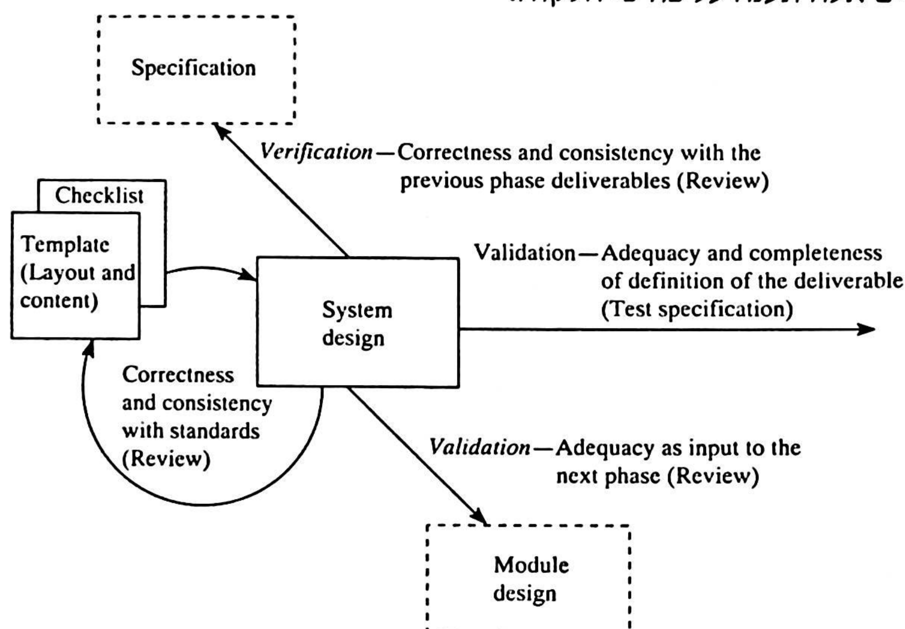
מחזור החיים, כפי שעיבדנו אותו באמצעות המתודולוגיה המתאימה, כולל עתה אוסף של שלבים מוגדרים היטב, כאשר אל כל אחד מהם מקושרת קבוצה של יחידות פלט מוגדרות ומתועדות. אם נשקול שלב של מחזור החיים במבודד, נמצא שיש לו קבוצה של מוצרים מוגמרים הזקוקים לאימות ולבדיקת תקפות (תרשים 4.4).

אימות המוצרים המוגמרים המתקבלים מאחד השלבים הוא בעיקרו סוג מסוים של **סקר** (review). בהנחה שיש צורך בעקביות של הקלטים לשלבים השונים, יתמקד סקר זה בהשוואה בין יחידות הפלט הנוצרות בשלב זה, לבין יחידות הפלט המתקבלות מהשלב הקודם - המוגדרות כקלט לשלב הנוכחי - ובכך שתוודא שהמוצרים המוגמרים יהיו תואמים לתקנים המוגדרים, כגון תבניות המסמכים ורשימות התיוג.



תרשים 4.4 המוצרים המופקים משלב הדרישות

ניתן להגיע לאימות גם בשלב מוקדם, על ידי כך שבוחנים אם המוצרים המוגמרים הנוכחיים מתאימים לשמש כקלט לשלב הבא, ואם המערכת שתיבנה בסופו של דבר ממוצרים אלה תענה על צורכי הלקוח.



תרשים 4.5 בדיקת המוצרים המופקים של שלב התיכון

ההיבט האחר של האימות הוא בעל חשיבות קריטית. ניתן להשיגו בצורה האפקטיבית ביותר על ידי הגדרת הבדיקות המתייחסות למוצר שיתקבל משלב זה, לכשהמערכת תיבנה לבסוף. במילים אחרות, אנו מביטים אל הצד הימני של תבנית-V ומגדירים את הבדיקות שנשתמש, לכשנגיע לנקודה זו. תרשים 4.5 מציג את משטר הבדיקות עבור מוצר אופייני של שלב בתיכון. זוהי דיסציפלינה מחמירה ביותר ורק לעיתים רחוקות ניתן לבצעה בפועל. לעיתים תכופות מושמעת הטענה נגד הניסיון לקבוע את מפרטי

הבדיקות כבר בשלב הזה, משום שרמת ההגדרה עדיין אינה מאפשרת את קביעת מפרט הבדיקות. ואם כן, האם ניתן להגדיר את התיכון בצורה שתאפשר להתקדם בביטחון אל שלב הפיתוח הבא?

יש תמיד מקום לעידון המפרטים ככל שמתקדם הפיתוח. עם זאת, העוסק בפיתוח וממשיך קדימה, מבלי לעגן את שלב הפיתוח על ידי קביעה מראש של מוצרי התיכון והגדרת הבדיקות הנלווים להם, מזמין לעצמו צרות בדמות מחלת הדרישות הזוחלות.

ניהול תצורה

ניהול תצורה (configuration management) הוא למעשה, הדבק המאחד את פיתוח התוכנה. אין קושי לעסוק בקטעי קוד המופעלים כל אחד בפני עצמו. לעומת זאת, המשימה נראית אחרת לגמרי כאשר יש צורך לקחת דרישה ולחולל ממנה מערכת אשר תכלול עשרות מודולים, שיפעלו בצורה נכונה, ויצטרפו בהצלחה ליצירת התוצאה הכוללת הרצויה.

הדיסציפלינה (ואכן יש כאן צורך בדיסציפלינה), של ניהול תצורה, משמעותה התבוננות קדימה אל הנקודה בה אולי תמצא את עצמך כשברשותך עשרות מודולים שונים, שלכל אחד מהם מספר גרסאות. אם תעשה זאת, תמצא את עצמך עסוק כל הזמן בסילוק שגיאות, ומהרהר בשאלה כיצד תדע איזו היא הגירסה הנכונה של כל אחד מהם שיש לצרף למוצר הסופי.

ניהול תצורה הינו תהליך בן שלושה חלקים, שלכל אחד מהם נודעת חשיבות עליונה:

1. **זיהוי** (identification). מתן שמות ייחודיים לכל פריט, וסימני זיהוי לגרסאות כדי שתוכל להבחין תמיד בין פריט לפריט.
2. **בקרת שינויים** (change control). מנגנון לבקרת שינויים, כדי להבטיח שכל מרכיבי הפעילות יהיו במסלול ובקצב הנכון. כשחל שינוי בפריט כלשהו, עלינו לוודא שכל הפריטים האחרים המתקשרים עם פריט זה, או משתמשים בו, ישונו אף הם (במידת הצורך) בצורה שתאפשר להם לפעול עם הפריט ששונה.
3. **יכולת מעקב** (traceability). מנגנון שיאפשר לנו לעקוב אחר הנתיב המוליך אל היווצרות כל פריט. למשל, במקרה שמשתנים מפרטי התיכון, עלינו לדעת מי הפריטים שנוצרו לפי מפרטי תיכון אלה, כך שנוכל לוודא שכל הפריטים ישקפו את המפרטים החדשים.

ניהול תצורה קשה ליישום ובלתי-אהוד, משום שהוא נוטה להאט את הקצב ולשים מכשולים בדרכם של מפתחים נמרצים. בדבר אחד אנו יכולים להיות בטוחים - כישלון בניהול התצורה, יביא לאסון במוקדם או במאוחר.

ניהול גרוע של התצורה יכול להתבטא בצורות רבות. ביניהן, מוצרים הנמסרים ללקוח ועדיין מכילים שגיאה שדווחה ונפתרה כבר לפני שלושה חודשים, התקנות הכושלות בשל העדר קובץ מסוים, או עדכונים הגורמים לנפילת מערכת, משום שתוכנית שנכללה בעדכון מנסה לגשת אל קובץ מערכת שתבנית הנתונים בו שונתה.

תחזוקת תוכנה

יש חשיבות לתחזוקת התוכנה, משום שהיא צורכת כמות משאבים גבוהה בהרבה מזו של פיתוח התוכנה. מוצר התוכנה הממוצע 'מבלה' רק 20 אחוז מחייו בפיתוח, שאר הזמן עובר בתחזוקה.

בדרך כלל, התחזוקה נחשבת כפחות משמעותית מאשר הפיתוח משום שמשימותיה, לרוב שינויים במערכות הקיימות, קטנות יחסית כאשר משווים אותן לפרויקטים הגדולים של הפיתוח. זוהי טעות חמורה, לא רק משום שהתחזוקה צורכת חלק גדול יותר יחסית של המשאבים שלנו, אלא גם משום שרבים משינויי התחזוקה הם קריטיים לתפקוד האפקטיבי של המערכת. העובדה שהשינוי מתבטא במספר קטן של שורות הקוד המושפעות ממנו אינה עושה את השינוי לבלתי חשוב, או לנטול סיכונים. הכללת שינויים בני ישורה אחת בלבדי בתוך השורות של קוד כלשהו עם מינימום של בדיקות או בכלל לא, אך מסתכמת בתוצאות הרות-אסון, היא תרחיש שכיח מדי.

לתחזוקה בעיות ודרישות מיוחדות משלה.

בעיות תחזוקת התוכנה

1. **חוסר תיעוד.**

התיעוד לא היה שלם בזמן האספקה וגם לא הושלם לאחר מכן, או שהתיעוד לא עמד בקצב השינויים. התוצאה היא שכל השינויים מעוצבים ברמת הקוד.

2. **בדיקות בלתי מספקות.**

הבדיקות מבוססות על גודל השינוי, במקום להיות מבוססות על הסיכון הכרוך בשגיאות תוכנה. שינויים קריטיים זוכים לאותו מאמץ שזוכים שינויים טריוויאליים.

3. **חוסר מתודולוגיה.**

שינויים קטנים אינם מצדיקים את התקורה של התיכון ושל בדיקות הביקורת. התוצאה היא שכל אחד 'עושה את החלק שלו'. שימוש שגוי בנתונים, ממשקי תוכנה לא נכונים וסגנונות תכנות ייחודיים, כל אלה תורמים להגברת הסיכויים לשגיאות.

4. **אין ניהול תצורה.**

הזמן הנדרש למעבר דרך נוהל בקרת השינויים גדול מדי עבור כל שינוי טריוויאלי. תוכניתן המבצע שינוי בתוכנית וחושף באג תוך כדי כך, סביר שיתקן אותו תוך כדי ביצוע השינוי. בינתיים, נפגם הממשק בין המודול 'המנופה' לבין מודול שנמצא במקום אחר.

5. **אין תכנון.**

כל דבר מתבצע 'תוך כדי תנועה', והתוצאה היא ששינויים נוטים להתנגש זה בזה.

התחזוקה מותנית כמעט לחלוטין בנהגי פיתוח טובים, שיאפשרו לוודא שפעילות התחזוקה תתחיל ממוצר מוגדר היטב, שהוא בעל תצורה מנוהלת וגם מתועד כהלכה. באותה מידה, חשוב שהמוצר יהיה פרי תיכון המביא בחשבון את התחזוקה. סיכוייו של מבנה מודולרי ברור עם ממשקים ברורים ופשוטים, להמשיך ולשמור על שלמותו, רבים יותר מאשר סיכוייו של 'כדור ספגטי'.

כיצד תומך תקן ISO 9000-3 ברעיונות מפתח אלה

הנחייה בנושא מחזור החיים

כל מבנה ISO 9000-3 מבוסס על העיקרון שפיתוח התוכנה יתנהל בהֶקשר מחזור חיים של פיתוח. בסעיף 5.1, מתבטאת ציפייה זו בקביעה שההתייחסות למחזור חיים, אין לפרשה כהפנייה אל מודל זה או אחר של מחזור החיים.

אם כן, תקן ISO 9000-3 מצפה מאתנו שנשתמש במודל מחזור חיים להגדרת המסגרת עבור נהגי הפיתוח. עם זאת, הוא אינו מנסה להכתיב מחזור חיים ספציפי, ואפילו אינו מציע ייעוץ לגבי מה שיכול להיחשב כמחזור חיים 'טוב'.

הנחייה בנושא מתודולוגיות

מתודולוגיות מופיעות בתקן ISO 9000-3 תחת שלוש כותרות נפרדות:

1. **שיטות וכלים לפיתוח** (5.4.2.3). הציפייה היא שתוכניות הפיתוח עבור פרויקטים יזהו את האמצעים שנועדו לוודא שהפעילויות מתבצעות בצורה נכונה. נוסף לכלים ולטכניקות, מאזכר סעיף זה כללים, נהגים ומוסכמות לפיתוח.
2. **תֶּכֶן ויישום** (5.6). מאוזכרות מתודולוגיות תיכון וגם מתודולוגיות יישום.
3. **פעילויות תומכות**. מאוזכרים **כללים, נהגים ומוסכמות** (6.5), **כלים וטכניקות** (6.6) מאוזכרים שנית, אך הפעם בהקשר של הארגון הכולל.

הקווים המנחים מספקים מסר חזק ביותר שיש לבחור ולהשתמש בכלים, טכניקות ושיטות על בסיס כלל-ארגוני. בעת התחלת פרויקט יש לתת את הדעת על הכלים, הטכניקות והשיטות עבור הפיתוח המסוים ולבחור אותם מבין אלה הזמינים בארגון. במקרה שכל אלה אינם זמינים או מתאימים, חייבת תוכנית הפיתוח לזהות גישה ספציפית לפרויקט בנושאים אלה.

זוהי גישה מובנית מאוד, אך אין לראות בה מרשם מחייב. הבחירה מסורה לחלוטין בידי המפתח, וזאת צריכה לכלול רק את הנוהג הקיים בתוך הארגון, בתנאי שיש בו הגדרה הולמת של התהליך. היתרונות הגדולים הנובעים מגישה זאת הם בדיוק אלה שקשורים בהגדרת מחזור החיים, כולם יודעים מהי הגישה שאושרה. יש להניח שלא כל אחד בצוות הפיתוח יהיה מאושר עם הגישה שנבחרה, אך מערכת מתועדת יכולה לשמש בסיס לדיון מושכל בדבר שיפורים שיוכנסו במערכת.

הנחיות בנושא אימות

תקן ISO 9000-3 מציג את נושא האימות בכל שלב בצורה חזקה ביותר (סעיף 5.4.6), וממליץ לערוך תוכנית לאימות פלט בכל שלב. קיימות גם גישות חלופיות לנושא האימות, אך הגישה של סקירות חוזרות של העיצוב מוצגת כגישה מתאימה.

ההגדרות של תקן ISO 9000-3 לאימות ולבדיקת התקפות הן כדלקמן

1. **אימות (לגבי תוכנה).**

   זהו התהליך של הערכת מוצרים (evaluating) שהופקו בשלב מסוים, כדי לוודא נכונות ועקביות של המוצרים והתקנים שהוזנו כקלט לשלב זה.

2. **בדיקת תקפות (לגבי תוכנה).**

   התהליך של הערכת התוכנה, כדי לוודא עמידה בדרישות מוגדרות.

הקווים המנחים מרחיקים אל מעבר לקביעת דיסציפלינה בסיסית. הם מצפים לכך שהתוצאות של פעילויות האימות ושל כל הפעולות שתינקטנה לתיקון שגיאות, תרשמנה ותיבדקנה בסוף כל פעילות. ואף למעלה מזה, הם מצפים לכך שרק המנות המאומתות של הפלט המיוצר על ידי הפיתוח, תוגשנה לניהול התצורה ותתקבלנה לשימוש מאוחר יותר.

הקווים המנחים ממליצים גם (5.4.4 ו- 5.4.5) שהדרישות לשלבי הפיתוח והתוצר של כל שלב פיתוח יוגדרו ויתועדו במלואם. המטרה העיקרית היא לאפשר אימות חיובי של פעילות השלב, אך גם כדי לוודא שמידע כגון קריטריונים לקבלה יועבר הלאה אל השלב המתאים במחזור החיים.

המקרה הפרטי של קלט השלב ההתחלתי - הדרישה עצמה - זוכה בצדק לאזכור מיוחד. אזכור קצר אבל חריף!

כל דרישה חייבת להיות מוגדרת בצורה כזאת שתאפשר לנו לאמת שהיא אכן הושגה.

ממשפט פשוט זה משתמעת הגדרה של קריטריוני קבלה מכומתים או ניתנים למדידה, או שתתקיים הדגמה אובייקטיבית שתוכיח את השגת הדרישה. למשפט זה מתוספת העצה שדרישות בלתי שלמות, מעורפלות או מנוגדות, תטופלנה עם הדרגים האחראים לניסות הדרישות.

הנחיות בנושא סקרים

סעיף 5.6.4 של תקן ISO 9000-3 ממליץ על סקרים כאמצעי שמוודה עמידה בדרישות, ושבודק אם שיטות העבודה שהוגדרו מתבצעות כהלכה. בהמשך לכך מופיע המשפט החשוב הבא:

אסור שהעיצוב או תהליך היישום יתקדמו כל עוד לא נפתרו בצורה משביעת רצון התוצאות של כל הפגמים הידועים, או כל עוד לא ידוע מהו הסיכון שבהמשך העבודה ללא פתרונות אלה.

עצה זו היא בעלת חשיבות רבה ביותר. לאחר כל ההמלצות בנושאי המתודולוגיה, הכללים וכל האמצעים האחרים שנועדו להגביר את חשיפת הסיכונים, מומלץ לשקול את הסיכונים שזוהו כבר, ולהחליט אם יש ביכולתנו להמשיך בביטחון בעבודה.

הנחיות בנושא בדיקות והוכחת התקפות

תקן ISO 9000-3 נותן ייעוץ כללי מסוים, בכך שהוא מציין שהבדיקות עשויות להידרש במספר רמות; אלא שבדיקת תקפות, בדיקות שדה ומבחני קבלה עשויים גם הם להיות חלק מאותה פעילות. קיימות מספר גישות שניתן לאמץ. כל גישה שנאמץ חייבת להיות מוגדרת בתוכנית בדיקות, שעשויה להיות חלק ממסמך אחר (כגון תוכנית איכות או תוכנית הפרויקט), או שתוכנית ה בדיקות תהיה מוגדרת במספר מסמכים (כגון תוכנית בדיקות של יחידת תכנות, תוכנית ליישום כוללת, ותוכנית למבחני קבלה). נראה שמרחב הפעולה גדול למדי.

בנקודה זו הופך תקן ISO 9000-3 להיות מאוד לא-מסייע! תחת הכותרת 'תכנון הבדיקה' (5.7.2) נאמר שהספק צריך ליצור ולבקר מסמכים שונים הקשורים בבדיקות. ניתנת שם רשימה של תחומים המחייבים תשומת לב. רשימת המסמכים כוללת מפרטי בדיקה, נהלים ותוכניות, אך אין הוא מחכים אותנו בכל הנוגע למה שצריך להיכלל במסמכים אלה. אנחנו חייבים אם כן, לבנות בעצמנו את הגדרות הפעולה שלנו.

הנחיות בנושא ניהול תצורה

תקן ISO 9000-3 מקדיש קטע שלם (6.1) **לניהול תצורה**, בו הוא מכסה את כל הסוגיות שנסקרו לעיל. קיימים ארבעה תחומים נוספים בהם יש חשיבות לניהול התצורה:

1. **בקרת מסמכים** (6.2) מהווה סוגיה בפני עצמה, אך בקרת העיצוב ותיעוד הניסויים קשורים קשר הדוק לבקרת השינויים.
2. לתחזוקה מוקדשים שני תחומי טיפול שבהם הזיהוי הוא קריטי: **זיהוי המצב ההתחילי** של המוצר (5.10.3) ונהלי המסירה, או השחרור של המוצר (5.10.7).
3. שכפול, הספקה והתקנה (5.9) כוללים בדיקות שמטרתן לוודא שהתיעוד הנכון אכן זמין. סעיף זה עוסק גם בצורך להחזיק עותקי גיבוי המשקפים את מצבו הנוכחי של המוצר.
4. סעיף הבדיקה (5.7.3) מציין שהשינויים יכולים להוליך לבדיקות חוזרים. כאן מתעוררות מספר סוגיות. השאלה בדבר מה שיש לבדוק בשנית ואיזה סוג של בדיקות חוזרות לבצע, תטופל מאוחר יותר. הסוגיה בה ניהול התצורה צריך לעסוק היא ששינויים בפריט תוכנה עשויים לחייב שינויים בבדיקות שנועדו לבדוק את תקפותו. לכן חייבים פריט התוכנה ובדיקתו להיות מקושרים בדרך כלשהי על ידי מערכת ניהול תצורה.

הנחיות בנושא תחזוקת התוכנה

סעיף 5.10 של הקווים המנחים דן בבקרה וברישום של **פעילויות התחזוקה**. למרות שהם מספקים הנחיות מסוימות בדבר מנגנוני הבקרה הבסיסיים, אין הקווים המנחים טורחים הרבה בהדגשת החשיבות של התחזוקה, או על יצירת הקישור עם הפיתוח.

נהלי פיתוח תוכנה

מחזור החיים הוא המפתח לזיהוי נהלי פיתוח התוכנה הדרושים לך. מתוך מודל זה של מחזור החיים ינבעו כל פעילויות האיכות שלך.

> ***ההחלטה בדבר הצורה הבסיסית של תהליך הפיתוח שתבחר בו, היא ההחלטה היחידה הגדולה ביותר שיהיה עליך להחליט אי פעם בהֶקשר למערכת איכות.***

אם אתה סבור שעבודת פיתוח תוכנה היא יותר מדי מגוונת מכדי שאפשר יהיה לתאר אותה במודל יחיד, אל דאגה! ייתכן שתזדקק ליותר מאשר מודל אחד, כלומר, מודל אחד לפיתוחים מובנים ומודל אחד לעריכת אבטיפוס. ייתכן גם שמודל אחד יוכל לענות על כל האפשרויות, אלא שאתה לא תהיה מעונין לכלול בו כל פעילות בכל פעם שתשתמש בו. יש מוצא עבור כל מגוון האפשרויות האלו. הדבר החיוני באמת הוא פגישה משותפת שלך ושל כל עובדיך כדי לסכם על תהליך הפיתוח. תרגיל זה להשגת הסכמה משותפת של קבוצת מומחי תוכנה לתהליך הפיתוח, יכול להפתיע במידת הקושי שבו. מאידך, תוכל למצוא עידוד בעובדה שככל שיגדל הקושי בקבלת ההסכמה, כך במרוצת הזמן, יגדל גם ערך המודל שייבחר.

מה הם הנהלים הדרושים לך? לאחר שתגדיר את מחזור החיים, יהיה עליך לזהות את תהליכי המפתח הכלולים בו ולתעדם בהתאם. ברמה המפורטת, יידרשו מספר כללים לתיכון וקידוד, יחד עם הנחיות בדבר השימוש באותן מתודולוגיות ותבניות שבחרת, לצורך יצירת מסמכי המפתח שמחזור החיים שלך מצפה שיופקו על ידי המפתחים. הנהלים יכולים לסייע בתיאור הגישה הכוללת אל האימות באמצעות בדיקות, ואל בדיקת התקפות באמצעות סקרים. עם זאת, בתחום זה, ארגונים רבים אינם חורגים מעבר למתן הנחיות בלבד, כשהם מניחים למנהלי הפרויקטים להגדיר בתוכנית האיכות את הגישה הנדרשת. ניתן גם להפיק נהלים לניהול פרויקטים שיהיו תואמים לתהליך פיתוח התוכנה.

מתודולוגיות מעשיות

המתודולוגיות קיימות כדי למלא שתי מטרות חיוניות:

- להקל על פיתוח המבוסס על צוות;
- להגדיל את שקיפות תהליך הפיתוח ולצמצם את הסיכונים.

ברור שבפרויקטים גדולים יש חשיבות לשני ההיבטים. במקרים כאלה, יש רק מידה מועטה של אי הסכמה בין המפתחים כל הוצאות התקורה הקשורות בשימוש במתודולוגיה מסוימת מתקבלות כהכרח הנדרש להגבלת הסיכונים למסגרת בעלת גבולות סבירים.

בפרויקטים קטנים יותר, בהם הצוותים בדרך כלל קטנים ומלוכדים, נראה תהליך הפיתוח כפחות בעייתי וניהול הסיכון הוא נראה כבעיה העיקרית. אין ספק שהתקשורת בתוך הצוות אמורה להיות קלה יותר וחשופה פחות לשגיאות מאשר בצוות גדול יותר, יחסית, גם קל יותר לטפח את התרבות המשותפת ואת צורת התיעוד. לכן, קשה יותר להצדיק בפני צוות הפיתוח את התקורה הכרוכה בשימוש במתודולוגיה, בעיקר את הזמן המוקדש לתיעוד. אף על פי שחלק גדול מהביקורת המוטחת כלפי המתודולוגיות מכוון לרוב נגד כל ניסיון להנהיג שיטת בקרה, אין להתעלם מכך שטמונה כאן בעיה אמיתית.

את המתודולוגיות יש לבחור עבור מטרות מסוימות. אכיפת מתודולוגיה שרירותית על פרויקט קטן יכולה לשמש סיבה לגיטימית לדאגה. עם זאת, יש בכל זאת יתרונות למתודולוגיות ואין להיפטר מהן בקלות. מה שנדרש כאן, הוא איזון זהיר בין היתרונות לבין התקורה. כך ניתן להגיע לסביבת פיתוח המצמצמת את הסיכונים ומגדילה עד למקסימום את ההזדמנות לעבודת צוות (במקרים המתאימים), מבלי לדכא את יוזמות התיכון הפתוחות לפני המפתחים. כיצד ניתן לעשות זאת?

ראשית, על ידי **סיווג הפרויקטים במונחים של גודל וסיכון**, כך שאפשר יהיה להגדיר את רמות ניהול האיכות עבור כל סוג וסוג, במקום להסתפק בהתייחסות כוללנית עבור כל הארגון. במסגרת סכימה כזאת, יכולים להיות לנו חמישה סוגים של פרויקטים:

סיווג הפרויקטים

1. **גדול ו/או יקר, שיש בו גורמי סיכון רבים.** למשל, למעלה משישה חודשי זמן פיתוח; צוות פרויקט גדול מארבעה איש; שימוש בטכנולוגיה חדשה.
2. **גודל בינוני או גדול, עם גורמי סיכון נמוכים יחסית.** למשל, שלושה עד שישה חודשי פיתוח עם צוות פרויקט של שלושה איש, או שישה חודשים אך ללא גורמים חדשניים.
3. **פרויקטים קטנים או בעלי סיכון נמוך.** למשל, פרויקט של איש אחד עד לשישה שבועות;
4. **משימות תחזוקה**, שיטופלו בצורה שונה מזו של פרויקטי פיתוח;
5. **פרויקטים פנימיים ליצירת אבטיפוס** למטרת הערכות, או הדגמות בלבד.

בעזרת סכימה ממין זה אנו יכולים להגדיר רמות תכנון, ותמיכת מתודולוגיה המתאימה לכל רמה. אנו גם יכולים להניח למנהלי פרויקטים לשפוט בעצמם לאיזה סוג משתייך הפרויקט שלהם.

שנית, אנו יכולים לנקוט בגישה מ'תוצרת בית' להגדרת המתודולוגיה, על ידי הגדרת כללים, קודי התנהגות, סטנדרטים, תבניות ורשימות תיוג. כל אלה צריכים להתאים לפעילויות שונות, שידוע שהן מכילות סיכון של שגיאות ממין מסוים. גם במקרה זה יש למנהלי הפרויקטים יכולת בחירה במתודולוגיה, שהפעם היא בנויה מ'ערכה' של רכיבים. נדון כאן בשתי דוגמאות של רכיבים כאלה.

- כללי תיכון;
- נהגי תכנות.

כללי תיכון

כללי תיכון המבוססים על היגיון פשוט יכולים לסייע בקביעת לנהגי תיכון טובים. הנחיות פשוטות בדבר פיצול פונקציות למודול תוכניות, הגודל האופטימלי לגודל תוכנית, מורכבות מבני נתונים, ותבנית להגדרת ממשקים. כל אלה ישמשו ליצירת יסודות בריאים. ערכי מדידה כגון **זיווג** (coupling), **לכידות** (cohesion) **ומורכבות** (complexity), יכולות לספק לנו מדידות כמותיות של התיכון. גם אם היעדים שייקבעו יהיו שרירותיים, רישומי הביצוע בפועל יאפשרו אופטימיזציית יעדים במרוצת הזמן. קווים מנחים מעשיים בדבר השגת יכולת הבדיקה, התחזוקה ויכולות מסוגים אחרים, יסייעו למעצבים למקד את תשומת הלב בסוגיות משמעותיות יותר של התיכון.

הצבת כללים כרוכה בסיכון מסוים. מה יקרה אם הכללים שגויים, ואנו נשיג פריון נמוך, או שאמינות המוצר תהיה ירודה? כל אלה מחזקים עוד יותר את הצורך בשיתוף מספר גדול ככל האפשר של מפתחים בדיון על הכללים, עוד לפני הפעלתם. משעה שנתחיל להשתמש בכללים אלה, ההשוואה בין מדידות ההצלחה שהושגה לבין הפרמטרים שנקבעו בכללים, תאפשר בטווח הארוך יותר להנהיג את ההתאמות הדרושות לאופטימיזציה של מידת אפקטיביות כללי התיכון.

נהגי תכנות

נהגי התכנות הם מקרה פרטי בתוך כללי התיכון, אך הם מתייחסים יותר לגילום הפיסי של התיכון, ולא לרעיונות המופשטים מאחוריו. יש חשיבות לצורה בה אנו כותבים תוכניות אלו. במקרים מסוימים החשיבות גבוהה יותר מאשר במקרים אחרים, אך היא קיימת תמיד.

ביישומים מסוימים מהירות הביצוע, או צמצום עד למינימום של השטח הנדרש לנתונים עשויים להוות שיקולים מרכזיים. ביישומים אחרים, היבטים של השפה שבשימוש עלולים להיות כרוכים בסיכונים. בכל המקרים, מי שצברו ניסיון בשפה וביישום יודעים מאיזה מכשולים עליהם להימנע וכיצד לעשות זאת. הם גם מכירים את השגיונות האופייניים לכל שפת תכנות ומהדר בהם הם משתמשים.

עדיף להעלות על הכתב את כל הידוע ולהביא אותו לידיעת עובדים מנוסים פחות, מאשר להניח לכל אחד ללמוד מתוך שגיאותיו. גם אם ידוע רק מעט יחסית אודות

השפה או היישום, היתרון שברישום הדברים הוא בכך שהם יכולים לשמש בסיס לתיקונים ולשיפורים במועד מאוחר יותר. החשוב מכל הוא שאין לאכוף מתודולוגיה מסוימת, ואפילו לא לזהות את סוג הכללים או התבניות שצריך להשתמש בהם, צריך לוודא שנשקלו כל האפשרויות ונשאלו השאלות המתאימות.

נהוג להגדיר כללים לצורת העריכה של הקוד, כדי לפשט את הכנת הסקר. כללים המתייחסים למורכבות המודולים יש בהם כדי להביא תועלת, משום שהם מסייעים לשמור את הבדיקות של המודולים בתוך גבולות סבירים. אפשר להשתמש בכללים אלה כדי להתריע על הצורך בבדיקות מיוחדות כאשר המעצב סבור שיש מקום לחרוג מהגבולות שנקבעו. לפעמים דרושים כללים הקובעים את השימוש במבני תכנות מיוחדים כדי להשיג תוצאות מסוימות (כגון הגישה העדיפה למבני לולאה), או איסור השימוש במבנים מיוחדים שהשימוש בהם עלול לערער את המערכת (כמו במערכת רגישה לבטיחות).

למעשה, כל אחד מאתנו חייב להפעיל את שיקול דעתו, כדי לקבוע מה הולם את הסביבה והיישום המסוימים שלו.

פיתוח המבוסס על טכנולוגיית 'הדור הרביעי'

טכנולוגיית הדור הרביעי מהווה מבחינות מסוימות גשר בין שיטות ידניות לשיטות ממוחשבות. היא מציעה נתיב 'קיצור דרך' אל מוצרי תוכנה, שבאפשרותו לחולל אוטומטית מוצרים מוגמרים של אחד, או יותר מהשלבים של מחזור החיים.

מוצרי הדור הרביעי, החל במחוללי קודים המפשטים את משימת התכנות, דרך מחוללי יישומים המחליפים כמעט לגמרי את שלב העיצוב הפיסי, מציעים יתרונות המחשוב במונחים של **הפריון** ההתחלתי, אך תוך **עלות** במונחים של ניהול המוצר לטווח הארוך.

שני מאפיינים מחייבים תשומת לב מיוחדת:

1. אי אפשר להשתמש בשלב שעבר הפשטה, או הוצא ממחזור החיים, כדי לחולל בדיקות של מוצרי התוכנה ההולכים ומתהווים. הכוונה היא, שכלי שממיר מודל של נתונים לקוד, מדלג על שלב העיצוב הפיסי מהדרג הגבוה. על ידי כך קשה, או שנמנעת האפשרות להגדיר בדיקות בארכיטקטורה הכוללת של התוכנה. האופי והממדים של הבעיה יבואו לידי ביטוי בדיון הבא בנושא הבדיקות.
2. לעיתים לא ניתן לשנות בנקל את המוצרים המוגמרים שמתחוללים אוטומטית. גם שינויים קטנים למדי יכולים לחייב אותנו לחולל מחדש את כל מערכת התוכנה.

פיתוח מבוסס חבילות יישומים סטנדרטיות או מותאמות

חבילות יישומים (application package) מייצגות יישומים שלמים הבנויים על פלטפורמה מסוימת של חומרה ותוכנה, כדי לענות על דרישות כוללניות בתחום של יישום מסוים. אפשר להשתמש בהן כמות שהן, או להתאים אותן התאמה אישית כדי שיענו על דרישות מוגדרות של הלקוח.

בעיות אופייניות לחבילות יישומים

1. לא סביר שהחבילה תענה בדיוק על דרישותיך, לכן עליך להחליט אם ברצונך להשלים עם הפערים, או להתאים את חבילת היישומים לצרכיך.
2. אם תחליט להתאים את החבילה, תצטרך להחליט אם לבצע בעצמך את העבודה (אם אתה יכול), או לבקש מהספק שיתאים אותה עבורך.
3. אם ספק החבילה מבצע את ההתאמות, יהיה עליך להחליט כיצד לאפיין את הדרישות, ולא פחות חשוב, כיצד ייערכו מבחני הקבלה. מבחן קבלה מלא מחייב מפרט מלא של הדרישות.
4. מבלי להתייחס לשאלה מי יבצע את ההתאמות, עליך להחליט איזה מהשינויים חיוני, ואיזה מהם אפשר לדחות או אפילו לוותר לגמרי. חשוב להיות פרגמטיים בתחום הזה, כדי להימנע משלב מורכב ומתמשך של פיתוח ובדיקה.

שיקולים אלה רחוקים מלהיות קלי ערך. אין לקבל בקלות החלטה לביצוע התאמות, ועליך להיות בטוח שהחבילה הסטנדרטית לא תוכל לענות על צרכיך. ההתאמות יכולות להיות קשות ואיטיות, ועדיין אין ביטחון שתקבל בדיוק את מה שאתה רוצה. אתה מקבל על עצמך את הסיכונים הנלווים לפיתוח, ורק יתרונות מועטים במקרה שתצליח. במקרה הטוב, תצטרך להשקיע מאמץ ניכר ביצירת תוכניות איכות ובדיקה כדי לוודא שאתה והספק שלך תגיעו להבנה בדבר מה שאתה מצפה שיסופק לך בתום מאמץ ההתאמה. מסוכן להניח שהחלקים שלא חל בהם שינוי יפעלו כולם כהלכה, ולא יהיו זקוקים לבדיקות.

ללא קשר להחלטתך בדבר ההתאמות, עליך לוודא שתבחר בתשומת לב רבה בחבילה הסטנדרטית, תוך שימוש בגישה כפי שמוצגת בקטע הבא.

הבחירה והשימוש בכלי תוכנה

השימוש **בכלי תוכנה** (software tools) מהווה מקור דאגה נכבד לכל מפתחי התוכנה. בקטגוריה זאת נכללים כלים שנועדו לסייע בתהליך הפיתוח (כלי CASE), כלים שנועדו לסייע בבדיקות (כלי CAST), וכלים שנועדו לסייע בניהול פרויקטים. כלי CASE ו-CAST משווקים בצורות רבות, ולא ייעשה כאן כל ניסיון לסווג או להעריך את כל הכלים הרבים המוצעים למכירה בשוק. מה שחשוב ביותר מנקודת הראות של האיכות הוא ש**הכלים שבשימוש יתאימו למשימה וייושמו בהצלחה**.

הערכה ובחירה של כלים

באיזה כלים נשתמש? רבים הארגונים שטעו בבחירה והתוצאה היתה שמשאבים יקרים לא נוצלו די הצורך, ובמספר מקרים, גם לא נוצלו כלל.

יש כמה עקרונות חשובים שיש לשקול בעת הבחירה. בראש וראשונה, חשוב לזכור שהסיבה העיקרית לקניית כלי תוכנה נועדה בדרך כלל להגדלת הפריון. זאת תושג על ידי מחשוב חלקים מתאימים של התהליך, ובמקרים רבים, על ידי הקטנת מספר השגיאות הנעשות תוך כדי תהליך.

כלי התוכנה יכול להיות שימושי בהגברת הפריון בדרך זו, אך ורק אם התהליך שאותו יש ברצוננו למכן הובן היטב ומיושם בצורה אפקטיבית. מחשוב תהליך שאינו מובן כהלכה מהווה סיכון חמור, משום שהוא מרשה לכלי להגדיר את התהליך. אנו עלולים למצוא שהצלחנו למחשב את התהליך, אלא שהתהליך אינו זה שהתכוונו אליו מלכתחילה. התוצאות יכולות לכלול עלויות הדרכה נוספות, ואפילו התנגשות בין תרבויות עבודה, משום שהתהליך החדש יהיה זר לשאר חלקי הארגון.

באותה מידה, התקנת כלי תוכנה לתמיכה בתהליך שאינו מיושם כהלכה, כמוה כהזמנה לבעלי התהליך להשתמש בכלי, כדי לתגבר את התהליך כפי שהם רואים אותו. התוצאה היא, שלעיתים תכופות נעשה בכלי שימוש שונה בחלקים שונים של הארגון, בדרכים שאינן תואמות כלל זו את זו. כלל הזהב הוא:

> ***החלט תחילה על התהליך, או על המתודולוגיה, ויישם אותם. לאחר מכן חפש דרכים לתגבר אותם באמצעות כלים.***

משעה שברורה המטרה לה נדרש הכלי, ביכולתנו להרכיב רשימה של מאפייני הדרישות שמולה נוכל להשוות את הכלים המוצעים ולהתחיל לחפש את הכלים שיכולים לענות על הדרישות. נזדקק גם למספר קריטריונים שכלי התוכנה יצטרך לעמוד בהם.

קריטריונים לבחירת כלים

1. מחיר.
2. מידת ההתאמה לסביבת החומרה המסוימת.
3. מתן תשובה לשאלות כגון:
   - ◊ באיזו קלות יוכלו העובדים ללמוד את השימוש בכלי?
   - ◊ כמה ארגונים אחרים משתמשים בכלי?
   - ◊ מהו המיקום של המשתמשים האחרים, בארץ או מחוצה לה?
   - ◊ מה דעתם של המשתמשים הנוכחיים על הכלי?
   - ◊ היכן ניתנת התמיכה לכלי, בארץ ומחוצה לה?

תשובות לשאלות אלו יש לרכז ולתרגם למדדים ניתנים לכימות, כגון:

4. העובדים צריכים להיות מסוגלים להשתמש בכלי לאחר הדרכה בת יומיים לכל היותר.
5. פיתוח הכלי והתמיכה חייבים להיות במדינת האם.
6. חייבים להיות לפחות עוד חמישה משתמשים אחרים.

הקריטריונים המכומתים חייבים להיות מסוכמים ומתועדים עוד לפני התחלת תהליך הבחירה, כדי שהבחירה תהיה אובייקטיבית. צפוי שתזדקק גם למספר משימות שהכלים המועמדים לבחירה יצטרכו לבצע. כך תוכל להשוות בצורה הוגנת בין הכלים. הקפד תמיד לבקש מהספקים להתקין את כלי התוכנה על החומרה שלך ולהריץ הרצה נסיונית. תוכל ללמוד רבות על הכלי ועל המפיצים במהלך שלב זה של תהליך הבחירה.

לאחר שתשלים את כל אלה, יוכלו הבדיקות והבחירה להתבצע בקלות רבה למדי.

ההכנה וההרצה של כלי התוכנה

בחירת כלי התוכנה אינה עדיין סוף המסע. אנו צריכים עדיין להתקין אותו וללמד את המשתמשים כיצד להפיק ממנו את כל היתרונות הצפויים. במקרה שעומד לרשותך תקציב נתון, **אל תיפול למלכודת** של הוצאת כל הכסף על כלי התוכנה עצמו.

זכור שההכנות לקליטת כלי תוכנה והרצתו, צפויות לעלות לא פחות מאשר הכלי עצמו.

מה כרוך בהכנות? ההדרכה היא הרכיב הראשון והבולט ביותר. אסור לקמץ בהדרכה - כלים משיגים את מלוא ערכם, רק כאשר המשתמשים חשים ביטחון בשימוש בהם ומכירים לפחות את רוב הדברים שכלי התוכנה יכול לבצע עבורם.

צפויות סוגיות נוספות שיחייבו טיפול. עלינו לוודא שהכלי פועל בעולם המציאות ולא רק במעבדת הבדיקה. דבר זה יהיה כרוך בהקמת פרויקט חלוץ כלשהו, כדי לסלק בעיות מוקדמות. נצטרך לקיים מרכז תמיכה זמני כל עוד המשתמשים רוכשים בקיאות בכלי. עלינו להביא בחשבון ירידה בפריון העובדים בשלבים המוקדמים של השימוש בכלי, ולהתאים את יעדינו בהתאם.

אם נשקול את כל הבעיות לפני שננסה להשתמש בכלי, תהיה הצלחתנו גדולה יותר כאשר ננסה להשתמש בו 'בשעות לחץ'.

ניהול הכלים

חשוב לזכור שבדרך כלל ניתן להשתמש בכלי התוכנה ביותר מאשר בדרך אחת. גם ייתכן במקרים מסוימים לקבוע תצורה שונה של הכלי, בצורה שתשנה את הפונקציונליות שלו. נתון זה מוסיף לעצמת הכלי, כל עוד השינויים יהיו מבוקרים.

עלינו להישמר מפני האפשרות שחלקים שונים של הארגון יפתחו נהלי שימוש שונים בכלי. זהו מתכון בטוח לכך שבפועל תהיה בידינו משפחה של כלים הקשורים זה לזה,

אך גם שונים זה מזה. לא רק שכתוצאה מכך תהיה התמיכה קשה ומורכבת יותר, אלא שייתכן שנאבד גם את היתרונות הנובעים מהפעלתו של כלי יחיד על פני כל הארגון.

תמיד יש מקום לשקול מינוי של 'מנהל כלים' ('tool manager') שיהיה אחראי לתמיכה בכלי ולפיתוח היישום שלו בדרך מובנית ועקבית. כך, שכל חלקי הארגון ייהנו מפירות השיפורים וכל השימושים בכלי יישארו תואמים זה לזה. מנהל כלים בקיא היטב בכלי יכול גם לשמש כזרז לשיפורים של השימוש בכלי.

פיתוח כלים בארגון

הרהור אחרון בנושא הכלים מתייחס לאותם כלים שאנחנו בונים עבור עצמנו, אולי משום שלא נמצא כלי מתאים אחר בשוק. לעיתים תכופות יהיו כלים אלה קטנים ופשוטים יחסית, אם כי לא תמיד.

אם אנו מודאגים בנושא איכות המוצר אותו שוקדים לפתח, עלינו לזהות את התרומה שיתרום הכלי למוצר המוגמר. אם הכלי הוא חלק חיוני של תהליך הפיתוח או ההספקה, עלינו להיות בטוחים שאינו מהווה חוליה חלשה בשרשרת האיכות שלנו.

עלינו לוודא שהכלי מתנהג בדיוק בהתאם למה שהתכוונו לו. כלומר, עלינו לאפיין את ההתנהגות הרצויה של הכלי, ולאחר מכן לבדוק את התקפות שלו מול אותו אפיון.

ניסוי

תכנון הבדיקה (test plan) או הניסוי שאנו מצפים או מתכוונים לבצע הוא ללא ספק האלמנט היחיד החשוב ביותר של תהליך הבדיקה. רבים התירוצים לאי-הכנה מוקדמת של תוכנית בדיקה. רוב התירוצים הם בנוסח של 'המוצר ישתנה ממילא עוד לפני שננסה אותו, אז לשם מה לתכנן?' אלה תירוצים כוזבים לחלוטין המונחים ביסוד חולשות האיכות שכולנו מצטערים עליהן.

תכנון בדיקה מאפשר לשקול בהקדם את אופי וממדי הבדיקה ההולמים את המוצר, או את המערכת שבפיתות. הדבר אינו מחייב ידע מפורט של התיכון, אך כן דורש הבנה באופי היישום שבפיתות, ושל הסכנות הנובעות מהפיתוח המסוים הזה.

כל הבדיקות צריכות להיות מותאמות לסיכון המשוער. הערכת הסיכון נמנת עם הפעילויות שקישרנו עם תכנון האיכות, והיא יכולה לשמש אותנו עכשיו כאשר אנו מתכננים את הבדיקות.

רמות הבדיקה

רצוי לשקול את ההכנות לבדיקה בשלוש רמות:

1. קביעת הדרישות ותוכניות הבדיקה כתגובה להערכת הסיכונים.
2. קביעת מפרטי הבדיקה על ידי הגדרת הבדיקות שיש צורך לבצע.
3. תיכון בפועל של הבדיקה.

בגישה זו, צריך להיות ברור שעד למועד סמוך לביצוע הבדיקות בפועל, איננו זקוקים לתיכון מפורט של הבדיקה. אפשר לדחות את העבודה המפורטת עד לאחר כתיבת התוכניות, ולצמצם בכך את הסיכויים לשינויים. אנו יכולים להרשות לעצמנו את הדחייה הזו, משום שמפרטי הבדיקה הגדירו כבר את אסטרטגיית הבדיקה במידת הפירוט המספיקה, כדי לקבוע שהמוצר המוגמר יענה על הדרישות. עם זאת, יש לזכור ש**אסטרטגיית הבדיקה** מונעת על ידי תוכניות הבדיקה והדרישות, המשקפות את הסיכונים הצפויים ואת התכונות הצפויות של המוצר.

תכנון הבדיקה

מה צריכה תוכנית הבדיקות לכלול? על התוכנית לקבוע את דרישות הבדיקה ואת הגישה שיש לנקוט כדי להשיגן. מבחינה זאת, דומה תכנון הבדיקה במובן הרחב לתכנון כולל של פרויקט. חשוב לזכור ש**תוכנית הבדיקות תצרוך כמחצית מהמשאבים ומהזמן הכולל של הפרויקט כולו**, ולכן הנושא ראוי גם לשיקול דעת מתאים.

עם גמר הכנת תוכנית בדיקה מחושבת היטב, מוגדרות גם פילוסופיית הבדיקה והקשר שלה לתהליך הייצור. בעוד שהפרטים עשויים להשתנות ובוודאי גם ישתנו, המסגרת מובטחת, משום שהיא קבועה כבר במדיניות. אם המדיניות לא תשתנה, דבר שיחייב את הסכמת קובע המדיניות, לא יוכלו ליהתמסמסי שרירותית מפרטי הבדיקה הבלתי מוגדרים עדיין. הכנת מפרטים אלה יכולה להדחות ללא חשש עד למועד שבו נגיע לרמת הפירוט הנכונה, ועד שתהליך הפיתוח יגיע לרמת יציבות שתצדיק את מידת תשומת הלב המפורטת יותר שנדרשת למפרטי בדיקה.

עם זאת, מן הראוי להשמיע כאן מילת אזהרה. אחת הסיבות השכיחות לדחייה בהכנת המפרטים והתיכונים של בדיקה, מקורה בעובדה שהמסמך שממנו יש לבנות את מפרטי הבדיקה אינו מכיל פירוט שיש בו כדי לקבוע מה צריכות הבדיקות לכסות. למרות זאת, תהליך התיכון נמשך לעיתים מאותו תיעוד שעליו נאמר שהוא מכיל פירוט בלתי מתאים.

> ***אם אין ברשותך מידע מספיק לבניית הניסוי, הרי שגם אין לך מידע שיאפשר להמשיך בעיצוב בביטחון.***

אם לא קיים מידע מספיק שיאפשר להחליט אם המוצרים שהוגדרו במפרט אכן הופקו בהצלחה (תפקיד מפרטי הבדיקה), אין ספק שיש מעט מדי מידע שאפשר יהיה לבסס עליו את המשך מאמץ הפיתוח. על ידי כך שנמשיך בתהליך התיכון לנוכח מצב זה, נצליח רק להעמיק את אי הוודאויות.

מה אמורה תוכנית בדיקה להכיל

לפנינו מספר נושאים שתוכנית בדיקה אמורה לעסוק בהם:

1. **דרישות הבדיקה**

דרישות הבדיקה אמורות להתייחס אל דרישות המערכת, או המוצר, ולעסוק בהיבטים של כל אחת מהן, כולל פירוט תכונות איכות רצויות, פריט אחר פריט. דרישות הבדיקה חייבות לעסוק גם בכל הסיכונים שאותרו ובמשמעותם לגבי הבדיקה. לעיתים תכופות יכולה טבלת מטריצה לשמש כאמצעי להפנייה צולבת שאפשר לסקור אותה באופן בלתי תלוי בכל הנוגע לשלמות. אפשר להשתמש בטבלאות קווי פעילות, שכבר הזכרנו קודם.

2. **מדיניות הבדיקה**

מתוך הדרישות לבדיקה ניתן להרכיב מסגרת כוללת. מסגרת זאת תציג את רמות הבדיקה השונות שנבחרו ואת הקשרים שביניהן. היא תזהה את אופי ומטרת הבדיקה שבכל רמה. המדיניות צריכה לקבוע את ממדי הבדיקה הנדרשים במידה שתקנה לנו את מידת הביטחון הנאותה, והיא יכולה להגדיר את רמת הכיסוי הנדרשת ברמה המבנית ו/או הפונקציונלית. כאן המקום לקביעת מדיניות התשתית הכוללת של הבדיקה; כלומר, דרישות לרישום התוצאות, שמירת הנתונים, השימוש בכלים או בטכניקות מסוימים, ומידת מעורבות המשתמשים.

3. **סביבת הבדיקה**

סביבת הבדיקה כוללת את פירוט תצורת החומרה והתוכנה עבור הבדיקה, את מנגנון בקרת השינויים, ואת הניהול הכולל.

4. **המשאבים לבדיקה**

משאבי הבדיקה עשויים לכלול כלים, אך אין ספק שהם יכללו גם עובדים. אומדן המאמץ הכולל הנדרש והכישורים הדרושים בכל שלב יאפשר לדאוג להדרכה מתאימה. תוכנית הבדיקה צריכה לזהות דרישות מיוחדות להדרכה ואת המועדים בהם תידרש הדרכה זו.

5. **ארגון ותחומי אחריות**

נוסף לדרישות הכוללות למשאבים, עלינו לדעת גם כיצד יאורגנו המשאבים. מי יהיה אחראי לאסטרטגיית הבדיקה הכוללת ולניהול כל רמת הבדיקה? כיצד ינוהל רישום השגיאות? מי יקבל את ההחלטות בדבר בדיקה חוזרת של תוכנה לאחר שיבוצעו בה התיקונים ובדבר הצורך בבדיקה רגרסיבית (ראה סעיף יבקרה ורישום של בדיקותי בהמשך)? מי יאשר את קבלת התוכנה בשלב הקבלה? מי יחליט אם להמשיך, או להשהות את הבדיקה כאשר מתגלות שגיאות? מי יאשר את הבדיקה החוזרת לאחר תיקון שגיאות?

6. **לוחות זמנים לבדיקה**

לאחר שיובאו בחשבון כל השיקולים הנזכרים, נוכל לתכנן את לוח הזמנים לבדיקה. לוח הזמנים צריך לכלול פעילויות מוקדמות כגון מפרטים ותיכון הבדיקה וכן את בדיקת המוצרים המוגמרים. תוכניות הבדיקה אמורות להצביע גם על

שיעורי הכשל הצפויים בכל רמה, כדי שניתן יהיה לאמוד את ממדי העיבודים החוזרים והבדיקות החוזרות, וגם להצביע על המשמעויות המוטבעות לתוך תוכנית הפרויקט הכוללת. לוחות הזמנים חייבים להיות מדויקים ככל האפשר במונחים של זמן ומאמצים שנדרשים. עם זאת, הם חייבים להיות מבוססים על אירועים, כגון הספקה של תצורות תוכנה מסוימות, ולא על תאריכים קלנדריים. על ידי כך, תישמר נכונות תוכנית הבדיקה על אף התוהו ובוהו שמסביבה.

מפרט הבדיקה ותיכון הביצוע

התיכון ומפרטי הבדיקה יימצאו כעת בפרספקטיבה הנכונה, כאשר נדון בפירוט המדיניות ויעדיה, כפי שנקבעו בתוכנית הבדיקה. במקרים רבים מתמזגים המפרטים והתיכון למסמך אחד, אך חשוב עדיין להפריד בין הנושאים שכל אלמנט עוסק בהם. מפרטי הבדיקה עוסקים במה שחייבים לנסות, בעוד תיכון הביצוע עוסק בדרך שבה יש לבצע את הבדיקה.

בשלב המפרטים יש צורך לטפל בשלוש סוגיות מפתח:

- מה הם יעדי הבדיקה?
- באילו טכניקות נשתמש בבדיקה?
- כמה בדיקות צריך לבצע?

התשובות לשאלות אלו אמורות להתקבל מהיעדים הכלליים ומהתוכנית, אלא שלגבי רמת בדיקה מסוימת עלינו לדון בהן בפירוט.

יעדי הבדיקה עונים על השאלה "למה": למה לנסות ברמה הזאת? מה הן הסוגיות המעסיקות אותנו? מה הדבר שברצוננו להדגים? היעדים צריכים להיות מפורטים במידה שתאפשר לזהות את הטכניקות המתאימות ואת הממדים הדרושים של הבדיקה. ברור שבדיקת כל אחד ואחד מהמוצרים, תהיה מטרה שלא ניתן להשיגה, ולכן לבחירה במדגם מייצג כבסיס לבדיקה תהיינה משמעויות לגבי האיכות (המדגם צריך להיות גדול במידה שתוכל להשרות תחושת ביטחון), והעלויות (מדגם גדול מהדרוש לא יגביר בהרבה את תחושת הביטחון, אם בכלל, אך וודאי יעלה יותר). קביעת יעדים מציאותיים תסייע בקביעת המדגם הנכון.

הטכניקות שנשתמש בהן מותנות במספר נושאים:

- אופי התוכנה שבוחנים, או מנסים;
- הכלים הזמינים;
- רמת הכיסוי הנדרשת;
- הטכניקה שנשתמש בה ברמות האחרות.

בסופו של דבר תצטרך הבחירה לוודא שיינתן הטיפול הנכון לסוגיות השייכות לרמה הזו, ושאיכות המוצרים שמרמה זו תתאים לשמש כקלט לרמה הבאה.

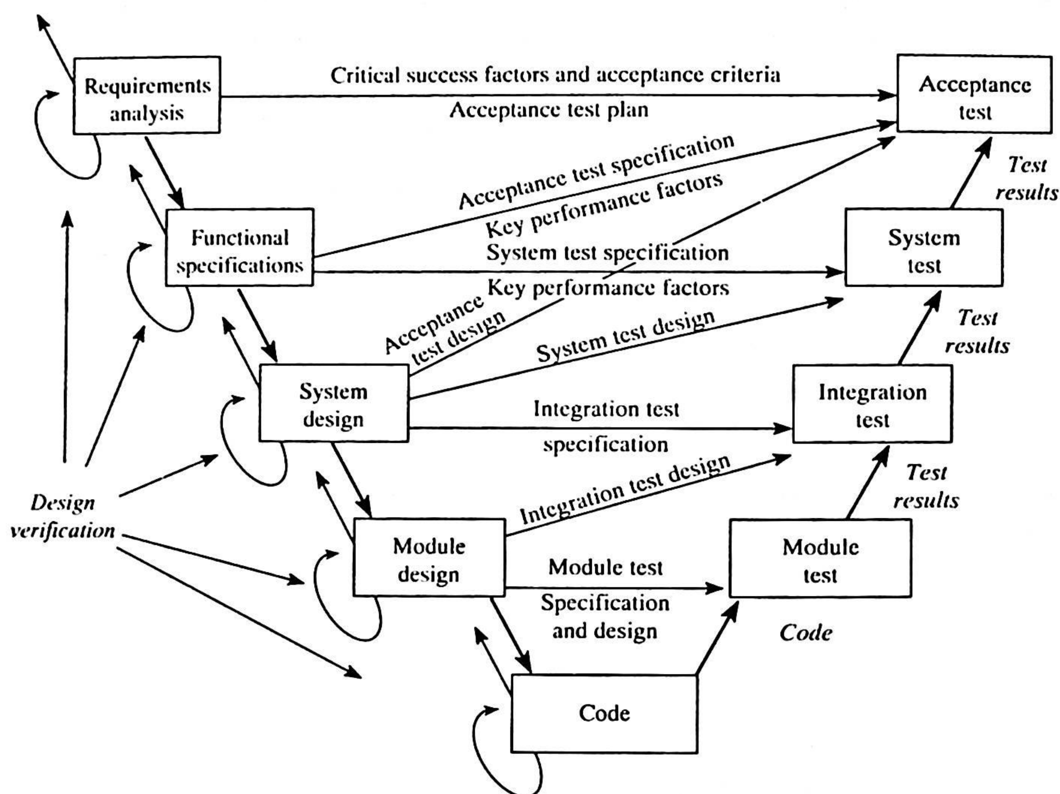
ממדי הבדיקה כרוכים במידה רבה בבחירת הטכניקות. מבלי להתחשב במדדים שנבחרו, חיוני לבצע כמות מספקת של בדיקות שיהיה בה כדי לוודא עמידה ביעדי הבדיקה. ניתן להגדיר זאת במונחים של כיסוי מבני (כגון, נבדקו 85 אחוז של כל ההחלטות), או של כיסוי פונקציונלי (כגון, הופעלו כל שגרות אימות הקלט), או שניתן להגדיר זאת במונחים של המשתמשים (כגון, יש לבדוק את התרחישים הבאים). בהעדר קביעה ברורה של ממדי הבדיקה הנדרשים להשגת היעדים, קיימת סכנה של קיצור הבדיקה במקרה שהפרויקט אינו עומד בלוח הזמנים. החלטה בדבר הכיסוי הדרוש, לא רק שהיא ממקדת את תשומת הלב על פונקציית הבדיקה, אלא שהיא גם מספקת קנה מידה מסוים לגבי הבדיקות שבוצעו בפועל, אם וכאשר יש צורך לקבל החלטה לקצץ באחד משלבי הבדיקה.

לעיתים מחליטים שאין יותר אפשרות להמשיך בבדיקה משום שהלקוח זועק לקבלת המערכת. לעיתים מפסיקים ב'אמצע הדרך', כאשר למשל הושלמו 43 אחוזים של הבדיקות הפונקציונליות וכי 38 אחוזים של הקוד טרם נבדקו. מובן שמנהל זכאי להחליט שיש לעצור את הבדיקה, אלא שכאשר הנתונים מכומתים ההחלטה מתקבלת תוך **ידיעה מלאה** או טובה יותר, של התוצאות הצפויות.

דרישה מכומתת לניסוי, מוליכה למדידת הישגי הבדיקה ולמידע מציאותי שניתן לבסס עליו החלטות בדבר הבדיקה.

לפני התחלת הבדיקה צריך להגדיר ולאשר את הנהלים הבסיסיים. מי שיבצעו את הבדיקה יזדקקו להוראות ברורות בדבר הדרך בה עליהם לבצע את הנהלים, כיצד לפרש את התוצאות, כיצד לרשום אותן ומה לעשות במקרה שהתוצאות לא יהיו משביעות רצון. את נהלי הבדיקות אפשר לפרסם כמסמך נפרד, או שבמקרים פשוטים ניתן לכלול אותם בתוך מסמכי הבדיקה.

ללא קשר לדרך שבה יושגו תכני הבדיקה המפורטים, יש תועלת רבה בהפקת מסמכי בדיקה שיכילו את כל המידע הרלוונטי עבור מבצע הבדיקה ומקום לרישום התוצאות, ובמקרה הצורך, גם דוגמאות של תבניות צפויות של הפלט. מסמכי הבדיקה מקלים על משימת הבדיקה ומצמצמים את החשיפה לשגיאות, הם גם מספקים את המכשיר ליצירת הרישומים הדרושים ללא צורך בהעתקה, או בשינוי צורת המידע. תרשים 4.6 מדגים כיצד מצטרפות פעילויות האימות ובדיקת התקפות ליצירת מחזור חיים כולל.



תרשים 4.6 מודל מחזור חיים לפיתוח תוכנה

בקרה ורישום של בדיקות

הבקרה של פעילות הבדיקה ניתנת להשגה בקלות רבה למדי באמצעות מסמכי הבדיקה. הפקה של מסמכי בדיקה משמשת אסמכתא לביצוע הבדיקה, ואילו החזרת מסמך הבדיקה למתאם הבדיקה המתאים, מהווה את ההשלמה של המשימה.

המעקב אחר המשמעויות של מקרי כשל יהיה בתחום אחריותו של המתאם, שחייב לוודא שהשגיאות דווחו ולשמש כמקשר עם צוות הפיתוח, כדי לוודא שהשגיאות אכן תתוקנה במסגרת של פרק זמן סביר. בשלב זה חשוב לוודא שייעשו זיהוי והערכה של כל ההשפעות התוצאתיות של שגיאות ותיקונים, על חלקים אחרים של המערכת.

המעקב אחר השפעת הבעיות וזיהוי ההשלכות הרחבות יותר של תיקונים שהתבצעו כבר, יהיו בדרך כלל, המשימה של צוות הפיתוח. עם זאת, האחריות לוודא שננקטו כל הפעולות המשלימות הדרושות, מוטלת על מתאם הבדיקה. זו דוגמה אופיינית לחשיבות שיש לממשקים ארגוניים וטכניים. מכאן משתמעת חשיבות הטיפול בממשקים אלה בתוכניות הפרויקט, ושל הצורך לוודא שתחומי האחריות יהיו ברורים בכל אחד מצדדי הממשק.

משעה שנעשה **ניתוח ההשפעות** (impact analysis), אפשר לקבוע את לוח הזמנים **לבדיקה משלימה** (follow-up testing). חזרה על בדיקה בה נמצאה שגיאה היא דרישה בסיסית, אך יש להניח שיהיה צורך בבדיקות רגרסיביות. בדיקות אלו נועדו לוודא שתחומים שלא היו אמורים להיות מושפעים על ידי שינויים ימשיכו להתנהג כראוי גם להבא. ייתכן שיהיה צורך לשנות תחומים המושפעים על ידי שינויים, ואלה עשויים להוליך לשינויים נוספים ולדרישה לשלבי בדיקה נוספים.

המשמעויות של שינויים יכולות להיות מרחיקות לכת. ניהול התצורה הוא רכיב חיוני בניסוי אפקטיבי.

בכל הנוגע לניהול הרישומים השונים, דרישה זאת אמורה להיות מוגדרת בתוכניות הבדיקה. תפקיד מתאם הבדיקות לוודא את הפקת הרישומים הדרושים. חלק מתהליך הרישום צריך לכלול תיעוד של תצורות החומרה והתוכנה שבשימוש בשלבי הבדיקה השונים.

לבסוף, משהושלמה הבדיקה יש לנסות ולהעריך את מידת ההתאמה ורלוונטיות הבדיקה. הדבר נעשה מתוך מגמה להשלים, במידת הצורך, את הבדיקות שהתבצעו עד עתה תוך מגמה להשיג את היעדים.

קבלת המוצר

סוף סוף מגיעים **לתהליך הקבלה** (acceptance process). לכל פעילות קבלה יש שני שותפים, הלקוח והספק. לכל אחד מהם תפקיד חיוני, ושיתוף הפעולה ביניהם הכרחי. לקבלה שלושה שלבים:

- הלקוח מגדיר את מדדי הקבלה.
- הספק מאמת את המוצר המוגמר.
- הלקוח מקבל את המוצר שנמסר לו.

הגדרת מדדי הקבלה היא אחת הסוגיות שצריך היה לדון בה במשותף בשלב ניתוח הדרישות. באופן אידיאלי, צריכים המדדים להיות מכומתים בכל מקום שהדבר מאפשר. במקרים שאי אפשר להגדיר מדדים מכומתים, יש לשאוף ליכולת הדגמה של המוצר. עם זאת, כאשר מגדירים אותם, צריכים המדדים להיות אובייקטיביים ולא סובייקטיביים. דרישה מהסוג שאומר: 'מסך בן 24 שורות שעומד בתקן אבג/123 של החברה, יחד עם אמצעי עזרה רגישים לטקסט' ניתן להדגים בצורה הולמת ואובייקטיבית; לא כן הדבר בדרישה מסוג 'ממשק ידידותי למשתמש'.

בדיקת תקפות המוצר השלם מחייבת את הספק לכלול רמת בדיקה שתהיה דומה לזו של הקבלה, בכך שמנסים את כל המערכת בתנאים הקרובים ככל האפשר לסביבה התפעולית. במספר מקרים, זוהי הבדיקה היחידה שמבצעת הספק, אך בדרך כלל זו השכבה העליונה של אסטרטגיית בדיקה מובנית.

פעילות הקבלה אמורה אף היא להיות מתוכננת. בדומה לקריטריוני הקבלה. שיטת ההערכה וסביבות התפעול של החומרה והתוכנה אמורות להיות מוסכמות זמן רב מראש. שיטת הטיפול בבעיות שמתגלות במשך הקבלה אמורה אף היא להיות כבר מוסכמת ומתועדת, אחרת יש להניח שיתעוררו ויכוחים ודחיות. ככל שמתקרבת פעילות הקבלה, אפשר להגדיר את לוחות הזמנים ואת המשאבים שיידרשו לניהול הבדיקה, ויש צורך בהסכמה משותפת לגביהם.

ניהול תצורה

בעיית יסוד בפיתוח תוכנה נעוצה בעובדה שהמוצרים בלתי נראים לעין, וניתנים לשינוי בקלות. להתקן אחסון המחזיק יחידת תוכנה אין שיטה אוטומטית להצגת תוכנו, והוא אינו מסוגל לשקף את העובדה שתוכנו השתנה. כתוצאה משתי עובדות לא נעימות אלו, מחייב הצורך בזיהוי ומעקב אחר התוכנה לאורך כל תהליך הפיתוח, נהלים מפורטים וממושמעים ביותר.

העונש על העדר בקרה על התוכנה תוך כדי הפיתוח יכול להתבטא בזאת שנמסור ללקוח את המוצר הלא נכון, או את הגירסה הלא נכונה של מוצר, או שהדבר יתבטא בהופעה חוזרת של כשל שתוקן כבר קודם לכן, או בהכללתה המקרית של פונקציונליות שפותחה בכלל עבור לקוח אחר ובהוצאות גבוהות. אלה אירועים מביכים ויש בהם פוטנציאל לתוצאות הרות אסון לנו וללקוחותינו. הדרך היחידה להתגונן מפני אלה היא בקרה קפדנית על כל הגרסאות של כל המוצרים מרגע יצירתם, לכל אורך חייהם עד לרגע מסירתם הסופית. תהליך זה נקרא **ניהול תצורה** (configuration management).

זיהוי ויכולת מעקב

הצעד הראשון ואולי החשוב ביותר של ניהול תצורה הוא הקמת מנגנון שיש בו קבוצת כללים מוגדרת היטב המשמשת לזיהוי הייחודי של כל אחד מרכיבי המערכת. כדי להבין מדוע, עלינו להתבונן קדימה בזמן לאותה נקודה בה נשקול ביצוע שינויים. אין זה משנה אם השינוי קורה לאחר המסירה הראשונית של המוצר, או במרוצת מחזור החיים של הפיתוח, התוצאות תהיינה דומות במידה רבה ומנגנון הבקרה יהיה זהה.

שינויים יכולים לקרות בשל מגוון סיבות, אך צפוי שתהיה להם אחת מהתוצאות הבאות: היישום הקיים יתוקן, או יורחב מבלי לשנות את הפונקציונליות, או את המטרות הבסיסיות, או שהשינוי ייושם כדי לחולל יישום חדש, או שונה משמעותית תוך שימוש ביישום המקורי כבסיס לחלק מהפונקציונליות הנדרשת. במקרה הראשון, אנו מתייחסים לשינוי כאל יצירת **גירסה חדשה** (new version); במקרה האחר, נשתמש במונח **עדכון** (variant).

במקרים בהם נדרש שינוי גירסה, אפשר לאתחל את השינוי בכל אחד משלבי מחזור החיים של הפיתוח. יכול להיות שינוי במפרטי הדרישות שיוליך לשינויים בתיכון ובקוד, וייתכן שיהיה זה שינוי ברמת הקוד שנועד לתקן השמטה מקרית. במקרה של

עדכון, חשוב לזהות את נקודת ההתחלה בה נוצר העדכון, ואת השינויים הקשורים ביצירתו; מנקודה זו ואילך ניתן לעקוב אחר גרסאות בעדכון עצמו, במקביל לגרסאות המוצר המקורי. בכל אחד ממקרים אלה, נצטרך לוודא מספר דברים חשובים:

- התיעוד שנוצר, החל מהדרישות ועד לקוד ישקף בצורה בהירה את המצב האמיתי של התוכנה הנמסרת.
- המסמכים והקוד יציינו בצורה בהירה איזו גירסה של איזה מסמך מתייחסת למצב הנוכחי של התוכנה הנמסרת.
- ניתן יהיה לאחזר במקרה הצורך את הגירסה הנוכחית וכל גירסה קודמת של התוכנה שנמסרה, תוך דרישה לזיהוי כל הכלים ותוכניות השירות ששימשו ליצירת גירסה כלשהי של התוכנה.
- **אירועי הבדיקה** (test cases) ורשומות הנתונים, משקפים בצורה נכונה את המצב הנוכחי של המערכת, תוך דרישה לזיהוי כל מידע הנוגע לבדיקה והגירסה או הגרסאות של התוכנה שנמסרה, במקרה שהדבר ישים לגביה.

קבוצת דרישות מחמירה זו אינה מותירה מקום לשאננות. במקרה שמתעורר צורך בשינוי ונעסוק רק בבעיית הזיהוי, לא נוכל להביא בחשבון את כל הבעיות האלו. חיוני להגדיר כבר מההתחלה את מנגנון הזיהוי, כך שבעת יצירת כל מסמך, תוכנית או **אירוע בדיקה** (test case), ניתן יהיה לזהותם בצורה ייחודית ולקבוע את הקשר שלהם לרכיבים המבוקרים האחרים.

אחת המטרות היסודיות של ניהול התצורה היא יכולת המעקב מכל דרישה אל המוצר המוגמר המתאים, כדי שנוכל לוודא שניתן מענה לכל דרישה. יכולת המעקב לאחור מהמוצר המוגמר אל הדרישה גם תאפשר לנו לוודא שכל מוצר מוגמר מקורו בדרישה מוגדרת (ולכן תצדיק גם את העלות והמאמץ שהושקעו בבניית יכולת זאת).

דבר אחרון שמנגנון הזיהוי מאפשר הוא הגדרת **מצב הבנייה** (build status), המשקף את מצבם הנוכחי של כל הרכיבים מהם מורכב המוצר. מצב בנייה יחיד משקף שילוב אוסף מסוים של רכיבים הנושאים מספר גירסה. אם נוכל להגדיר באותה עת צירוף מסוים של רכיבים מבוקרים המהווים במשותף גירסה מסוימת של המוצר בכללותו, נוכל לזהות את מצב הבנייה הנדרש בעת מסירת המוצר. הפער בין מצב הבנייה הנוכחי לבין מצב הבנייה הנדרש לאספקה, יגדיר את תוכן התוכנית לבנייה ולאספקת המוצר.

סוגיות זיהוי אלו הן הבסיס לבקרה, המנגנון הבסיסי חייב להיות מטופל בתוכנית לניהול התצורה (אם קיימת כזו), בתוכנית האיכות, או בתוכנית הפרויקט.

בקרת שינויים

בקרת השינויים (change control) בפריטי תוכנה היא אבן הפינה בניהול התצורה. להגדרות הזיהוי הייחודיות שנקצה לפריטים אלה תהיה משמעות רק אם נבקר את תהליך הצירוף של כל פריט חדש, גירסה או עדכון. כך נוכל לדעת איזה מהם אושר אחרון. כדי שהדבר אכן יקרה, לא מספיק לתת לגירסה האחרונה את המזהה הייחודי שלה; עלינו לוודא גם שכל מי שמשתמש בגירסה ידע שחל בה שינוי ושייתנן לו עדכון

כתחליף לגירסה הקודמת. רק על ידי דיסציפלינה מסוג זה נוכל לוודא שבמשך תהליך הפיתוח תימצא בידי כל אחד הגירסה האחרונה.

חובה לדאוג לכך שתהליך בקרת השינויים יתבצע כהלכה: מעט מדי או מאוחר מדי פירושם אסון פוטנציאלי, יותר מדי או מוקדם מדי מוליכים לחנק תהליך הפיתוח.

מהו הזמן הנכון בו נפעיל בקרה על התוכנה? הכלל הפשוט ביותר קובע שיש לדחות כל צורת בקרה עד לשלב שבו יש ליותר מאיש אחד צורך בגישה אל פריט. החל מנקודה זאת קיימת סכנה אמיתית במקרה שיבוצעו שינויים בצורה בלתי מבוקרת. לפני אתחול הבקרה תצטרך להיות צורה כלשהי של מנגנון מאשר, כדי לוודא שהפריט נמצא במצב שמאפשר את השיתוף בו, כלומר, עריכת סקר. תוצאת הסקר תהיה מסמך, או פריט תוכנה במצב ידוע (כפי שדווח בעת הסקר), ומצב ידוע זה יוגדר כגירסה מספר 1.

לעיתים תכופות יהיה צורך להשתמש בצורת בקרה כלשהי עוד לפני נקודת בקרה זו. ניתן להשיגה באמצעות גרסאות טיוטה, תוך שימוש בסדרה נפרדת של סקרים. טיוטת סקר A תשקף מצב מוגדר אך לא בהכרח תעבור את תהליכי הסקר והאישור. שינויים במסמך כזה יכולים להימשך בצורה לא רשמית תוך התגבשות המסמך, משום שבכל מקרה לפני המהדורה הרשמית יבוא סקר רשמי בו יוגדר מצב המסמך במלואו.

משעה שמופעלת בקרת שינויים, תתחיל האבטחה הפיסית להוות שיקול רציני. את המהדורה הרשמית הראשונה יהיה צורך לשכן בספרייה נפרדת, וכל המהדורות העתידיות יצטרכו להעשות רק באמצעות הספרייה. אם נקפיד על כך שגרסת הספרייה תוכל להתעדכן רק דרך בקרת השינויים הרשמית, נוכל להיות בטוחים שכל מי שיבקש מהספרייה עותק של המסמך, יקבל תמיד את הגירסה המאוחרת ביותר.

במקרה שמספר אנשי צוות עובדים בו-זמנית על אותו פריט תוכנה, הספרייה תוודא שכל העדכונים מסונכרנים בדרך הנכונה. במקרה ופריט מהווה חלק מתוך יותר מאשר מוצר אחד, הספרייה תתאם את עדכון כל המוצרים המושפעים מהשינויים. הספרייה תצטרך גם לנהל רישום של המחזיקים בעותקים של כל פריט או מסמך, כדי לוודא שיקבלו הודעה על עדכון. אחזקת רישום מחזיקי מסמכים, או פריטי תוכנה תהיה אם כן, משימה המוטלת על הספרייה.

תהליך בקרת השינויים עצמו חייב להיות בעל מספר מאפיינים חשובים. מאפיין ראשון הוא האמצעים לאתחול שינוי, כגון טופס דרישה לשינוי שיהיה נגיש לכל אלה שיכולים לחולל שינויים. דרישות אלו יצטרכו להיבדק לפני אישור השינוי, כדי לוודא שהשינויים המוצעים ישימים ורצויים. לאחר אישור השינוי נצטרך לדעת את משמעויותיו במונחי ההשפעה שתהיה לו על מוצר או מוצרים שהפריט המועמד לשינוי הוא חלק מהם. ניתוח ההשפעות ישאף לזהות את כל ההשפעות הנובעות משינוי נתון. אם ניתוח ההשפעות אינו מגלה תופעות לוואי בלתי רצויות, ניתן להתחיל ביישום השינוי ובבדיקתו. הבדיקה תצטרך לכסות לא רק את השינוי הפונקציונלי שנקבע, אלא גם להיות בטוחים שהפונקציונליות הנשארת תמשיך להתנהג כפי שמצפים ממנה. צורה זו של בדיקות רגרסיביות צריכה להיות מיושמת גם לכל התחומים שזוהו על ידי ניתוח ההשפעות כתחומים שצפויים להיות מושפעים מהשינוי.

לאחר שנעבור בהצלחה מכשולים אלה, ניתן יהיה להכניס את השינוי לספרייה ולהודיע על כך לכל מי שדרוש. יש לעדכן את רישומי הספרייה כדי שייראו הקשרים בין הגירסה החדשה לבין הדרישה לשינוי שחוללה את השינוי, או את השינויים.

לבסוף, הספרייה תזדקק ליכולת להפיק דוחות תצורה שיזהו את מצב פריטי התוכנה ויעקבו אחר השינויים שאירעו מאז כניסתם לראשונה למערכת הבקרה.

תכנון ניהול תצורה

רבות מהסוגיות שנידונו תהיינה בעלות משמעות כללית ולכן תצטרכנה להוות חלק מהמדיניות הכוללת של הארגון לניהול התצורה. עם זאת, יש תחומים שיצטרכו להיות מוגדרים עבור כל פרויקט:

- ארגון ותחומי אחריות, כגון מי יהיה אחראי לאשר שינויים ומי יהיה הספרן;
- פעולות ניהול התצורה שיש לבצע, כגון בקרת השינויים;
- כלים, טכניקות ושיטות בהן יש להשתמש;
- השלב בו יש לכלול פריטים במסגרת בקרת התצורה

כיצד משתלב תהליך הפיתוח הבסיסי במערכת ניהול איכות (QMS) שלנו

בפרק זה בחנו אחדות ממשמעויות תהליך הפיתוח הבסיסי שעלינו להיות מודעים להן דרך קבע, בשל הסכנות הפנימיות הנובעות מטיפול בדבר כה תובעני כמו פרויקט ניהול תוכנה. עם זאת, התהליך הבסיסי חייב להיות המוקד הראשוני שלנו.

תהליך הפיתוח הבסיסי חייב להיצמד לשמונת עקרונות הנדסת התוכנה שהגדרנו בפרק 1, וזאת דאגתנו הראשונה. האלמנטים המרכזיים של עקרונות אלה:

אלמנטים מרכזיים של תהליך פיתוח

---

1. **הגדרות של:**
   - מחזור חיים של פיתוח תוכנה;
   - מתודולוגיה;
   - אסטרטגיה לאימות ובדיקת תקפות;
   - דיסציפלינה לניהול תצורה.
2. **נהלים לפיתוח תוכנה**

נהלים אלה מתארים את התהליכים היסודיים עבור חלק זה של העסק, עליהם לכלול הנחיות בדבר בניית המפרטים.

3. **תכנון פעילויות הפיתוח**

תוכנית הפיתוח מגלמת את לוח הזמנים לפעילויות הפיתוח, המבוסס על מודל מחזור החיים והמתודולוגיה שנבחרו עבור הפרויקט. לכן, מסגרת הפיתוח חייבת להיות קיימת עוד לפני שניגשים לתכנון הפרויקט.

4. **תכנון פעילויות האיכות**

פעילויות האיכות שהוגדרו בתוכנית האיכות מסייעות לבלימת הסיכונים הכרוכים בפרויקט, הן חייבות לכלול את אסטרטגיית האימות ובדיקת התקפות הייחודיות לפרויקט זה, כולל הדרישות לבדיקה חוזרת של התיכון ולתכנון הבדיקה.

5. **כללים וסטנדרטים**

תהליכים יצירתיים כדוגמת התיכון והתכנות אינם יכולים להיות כבולים בתוך מסגרת של נהלים, אך הנחיה ונהגים טובים יסייעו לשפר את העקביות ולפשט את קיום הסטנדרטים.

6. **תבניות למסמכים**

מתארים של מסמכי מפתח, המראים את המתאר הדרוש ומגדירים את התוכן הצפוי מקלים על עריכת מסמכים ובדיקתם האפקטיבית.

7. **כללי התנהגות**

בתחום בו הנהגים נמצאים עדיין בשלבי התפתחות ושיפור, יש תועלת בתיעוד המצב הנוכחי של הפיתוח, כדי לקדם עקביות מסוימת ולעודד ויכוחים בנושא השיפורים.

8. **מנגנון לבקרת התצורה**

דיסציפלינה לניהול התצורה מחייבת יישום מעשי באמצעות כללים לזיהוי, ליכולת המעקב ולבקרת השינויים. יש צורך בקיום ספרייה שתאפשר בקרת תפוצה של מסמכים ותוכנה.

9. **אסטרטגיית המדידה**

התחזוקה והשיפור של תהליך פיתוח התוכנה הבסיסי יזדקקו למידע כמותי בדבר הביצועים הנוכחיים. אסטרטגיית המדידה תזהה את המדידות הדרושות ותגדיר כיצד לאסוף, לנתח ולשמור אותן.

כעת, יש ברשותנו שני נתיבים אינטראקטיביים במערכת האיכות: ניהול הפרויקט ופיתוח התוכנה. עלינו לנהל ולשפר פעילויות הדדיות אלו בכך שנשכן את כל המבנה בתוך סביבה ניהולית תומכת ומעוררת, כדי שלא נסתפק בקיום התקנים בלבד, אלא גם נשתדל לשפרם. זאת תהיה המשימה של מערכת איכות, בה נדון כעת.

פרק 5

עיקרי מערכת האיכות - לב המערכת

מבוא

מערכת איכות חייבת להיות ערוכה להישרדות ולשיפורים מתמידים. כאשר נערמים הקשיים, ההישרדות מחייבת שמערכת האיכות לא תהיה הקורבן הראשון, והיא צריכה להיות איתנה די הצורך לעמוד בלחצים של הטווח הקצר. השיפורים המתמידים חייבים להוות מנגנון מובנה שיספק למערכת מידע בדבר ביצועיה, ושעל פיו ניתן יהיה לקבוע את המדיניות לעתיד. יש בכך כדי לוודא את ההתמדה במאמץ האיכות. לגבי תקן ISO 9000-3, תחזוקת המערכת היא הדרישה הראשית, אולם בחיפושינו אחר תמורה ארוכת-טווח לכספנו, אנו חייבים להתבונן אל מעבר לתחזוקה, באמצעות תוכנית שיפורים מתוכננת. פרק זה מאגד מספר נושאים שטופלו על ידי תקן ISO 9000-3 ומצעיד אותם הלאה. מטרת פרק זה לזהות ולחקור היבטים של מערכות האיכות, שתרומתם רבה ביותר לתחזוקה ולשיפורים לטווח הארוך.

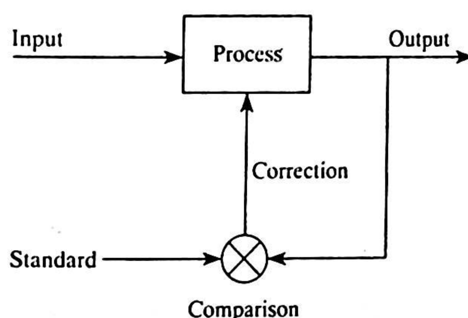
מהו עיקר מערכת איכות

מנגנוני האיכות שזיהינו עד כה התמקדו בתהליך 'הינדוסי התוכנה' (מלשון software engineering). בעוד שדיסציפלינות אלו חיוניות ויסודיות למערכת איכות תוכנה אפקטיבית, כשלעצמן אין בהן כדי להבטיח את המשך האפקטיביות של מערכת האיכות.

הדיסציפלינות, ככל שתהיינה מבוססות ונתמכות היטב, כמעט תמיד נוטות להתפורר במרוצת הזמן. שינויים בהרכב העובדים מביאים לשינויים בנקודת המבט ויש בהם כדי להוליך לסחף הדרגתי במחויבות המשותפת שסיפקה את הדחף ההתחלתי ליישום הדיסציפלינות. שינויים בסביבת מערכת האיכות נוטים גם הם לצמצם את אפקטיביות הדיסציפלינות האלו. לקוחות חדשים ורעיונות בדבר מוצרים חדשים עשויים לחייב גישות חדשות. דיסציפלינות שהיו חיוניות בעבר, עלולות להפוך במרוצת הזמן למנוגדות לפריון.

כיצד נוכל לשמור על המשך קיומה של מערכת איכות לנוכח שינויים מבפנים, ועם זאת לאפשר לה לשנות את עצמה, כדי להיענות לשינויים שמבחוץ? יציבות וגמישות הן תכונות הסותרות זו את זו. המנגנון שנפעיל להשגתן יצרוך אף הוא משאבים, ועל ידי כך יפגום בפריון הכולל.

התשובה לכך היא ביצירת מצב יציב המלווה בגישוש אחר שינויים, פנימיים וחיצוניים, ונקיטת פעולה מתאימה בתגובה לכל שינוי שנחוש. גישה זו מונחת ביסוד בקרת הלולאה הסגורה, כפי שהיא מוצגת בתרשים 5.1.



__תרשים 5.1__ בקרה בלולאה סגורה

פעילויות עיקריות במערכת האיכות

מערכות איכות מתוחזקות על ידי מערכת בקרה בלולאה סגורה. פירוש הדבר, שעליהן להיות מסוגלות לקבוע יעדים, לחוש בסטיות, ולשנות את התנהגות המערכת כדי להחזירה לכיוון היעד.

במערכת איכות, התנהגות על פי יעדים נקבעת על ידי הגדרת הדרך בה צריכה המערכת לפעול. עלינו להתייחס לשלושה היבטים:

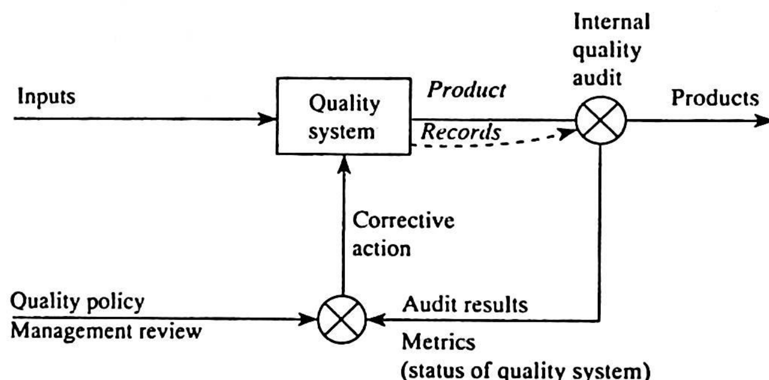
- **מדיניות האיכות** מגדירה את מטרות האיכות הכוללניות;
- **נהלי האיכות** מפרטים את המדיניות ומגדירים את הדרכים להשגתה;
- **תפקידים ותחומי אחריות** מגדירים את הדרך בה צריכים העובדים להפעיל את המערכת.

כל אחת מההגדרות צריכה להיות מתועדת פורמלית ולהיתמך במתן הדרכה, כדי לוודא את יישומן העקבי.

ההתנהגות בפועל של המערכת צריכה להיות נתונה למעקב על ידי תהליך ניטור שיוודא גילוי סטיות מההתנהגות הרצויה ותיקונן בהזדמנות הראשונה. דבר זה מצריך:

- **מבדקי איכות** (quality audits), שיספקו נקודת מבט עצמאית על דרך פעולת מערכת האיכות ויזהו סטיות מהמערכת המתועדת ברמה התפעולית;

- **מדידה** (measurement), שתאסוף מצביעי ביצועים עיקריים, גם של המוצרים וגם של התהליכים;
- **מנגנונים לפעולה מתקנת** (corrective action mechanisms), שיוודאו את הטיפול בבעיות, בזמן וביעילות;
- **סקרי הנהלה** (management reviews), שיסקרו דרך קבע את מערכת האיכות ואת תהליך הניטור, כדי לוודא טיפול בסוגיות רחבות יותר של המדיניות.



תרשים 5.2 בקרה בלולאה הסגורה של מערכת איכות

כל הפעולות האלו שהן לב המערכת, עולות כסף וצורכות משאבים שניתן היה להפנותם לייצור. לכן, על ההנהלה להיות בטוחה שיש אמנם צורך בפעילויות האיכות וכי הן מתבצעות בדרך האפקטיבית ביותר. מחויבות ההנהלה חיונית לפעולה האפקטיבית ולהמשך הישרדות מערכת האיכות.

מערכת איכות אינה יכולה להישרד ללא מחויבות ברורה של ההנהלה.

אין בעולם האיכות דבר שהוא חשוב ממחויבות ההנהלה. גם אם נצליח להקים מערכת פעילה בטווח הקצר, הרי שבלי מעורבות פעילה של ההנהלה, קלושים סיכויי ההישרדות של מערכת זו מעבר למשבר הראשון.

מחויבות ההנהלה יכולה להתבטא בשלוש דרכים:

- על ידי **הודעה פומבית** בדבר גישת הארגון לאיכות במסגרת מדיניות האיכות;
- על ידי **הקצאה מחייבת של משאבים** להשגת מדיניות האיכות;
- על ידי **סקרים סדירים** של ביצועי מערכת האיכות ונקיטת הפעולה הנדרשת לתיקון הבעיות שתתגלנה.

נבחן כל אחד מרכיבים אלה על ידי כך שנתבונן במבנה מערכת הבקרה בלולאה סגורה, המהווה חלק מדרישות תקן ISO 9001.

תקן ISO 9000-3 ומה שמעבר לו - עיקרי מערכת האיכות

הקווים המנחים של תקן ISO 9000-3 משקפים בדייקנות את דרישות תקן ISO 9001 בכל הנוגע לפעילויות העיקריות. זה מרמז על גישה שונה במקצת מזו של חלקי הנדסת התוכנה של המערכת, בהם הקווים המנחים מאפשרים לנו מרחב תמרון מסוים. בתחום העיקרי, לב מערכת האיכות, 'הקווים המנחים' משקפים סעיפי חובה שבתקן ISO 9001, שאנו חייבים לעמוד בהם, אם ברצוננו למלא אחר דרישות התקן.

בפרק זה נחקור את פעילויות לב מערכת האיכות בשלוש דרכים:

- על ידי הצגה והסבר פעילויות לב מערכת האיכות המוגדרות בתקן ISO 9000-3;
- על ידי תיאור מספר פעילויות נוספות (שהוגדרו בתקן ISO 9000-3 כ'פעולות תמיכה') שאפשר לראותן כחלק של לב המערכת.
- על ידי הצעת גישה ליישום לב מערכת האיכות, המתבוננת מעבר לקווים המנחים המיידיים של תקן ISO 9000-3 לכיוון אסטרטגיה של תהליך שיפור לטווח הארוך.

התחומים שהוגדרו בתקן ISO 9000-3 כחלק של לב מערכת האיכות הם:

- תחומי אחריות ההנהלה, הכוללים:
  - ◊ מדיניות האיכות;
  - ◊ ארגון ותחומי אחריות;
  - ◊ נציג ההנהלה;
  - ◊ סקרי הנהלה;
- תיעוד מערכת האיכות;
- מבדקים פנימיים (audits) של מערכת האיכות;
- פעולות מתקנות.

נעסוק גם בארבע פעילויות הקשורות לנושא:

- בקרת תיעוד;
- רשומות איכות;
- הדרכה;
- מדדים.

הגישה בפרק זה שונה במקצת מזו שבשני הפרקים הקודמים, בכך שהדיון בתקן ISO 9000-3 ובהנחיות המעשיות של היישום, מוצגים יחד.

מדיניות האיכות

תקן ISO 9001 מחייב להגדיר ולתעד את מדיניות ויעדי האיכות, ובכלל זה הכרזה בדבר מחויבות הארגון לאיכות, לוודא שדברים אלה יובנו, ייושמו וגם יקוימו בכל הדרגים שבארגון. לעיתים, מתבצעת חובה זו על ידי הכרזת מדיניות קצרה הנכללת בחוברת האיכות, הצהרה המתייחסת לפעולה על פי תקני ISO 9001 ו- ISO 9000-3. עם זאת, אפשר לעשות יותר מזאת כדי **לחולל תחושה אמיתית של מחויבות**.

כל ארגון זקוק ליעדים. בלעדיהם לא יהיה לו בסיס למדידת ביצועיו או להבעת כוונותיו לעתיד. **יעדי איכות** קובעים את המטרות להשגת האיכות ומתארים את תרומתה הצפויה של האיכות להשגת היעדים הכלליים.

מדיניות איכות מגדירה את הגישה הכוללת לאיכות ומזהה את הדרך בה יש להשיג את יעדי האיכות. במסגרת מדיניות האיכות תיכלל הודעה על מה שמצפים מעובדי החברה והודעה בדבר צעדים שהחברה תנקוט בהם, כדי לתמוך בעובדיה להשגת יעדי האיכות. האלמנט האחרון הוא ביסודו הודעה בדבר מחויבות הארגון לאיכות.

הכרזה ברורה בדבר מדיניות האיכות חובקת את כל האלמנטים הנזכרים, בהכרזה ברורה ופשוטה המתווה את השותפות שבין הארגון לעובדיו, ומכוונת להפיק את מירב הביצועים לתועלת הכול. טוב יותר יהיה אם מדיניות האיכות אף תזהה את המדידות הדרושות כדי להציג את ההתקדמות לעבר יעדי איכות. תפקיד מסמך האיכות להרחיב בנושא מדיניות האיכות ושאר חלקי מערכת האיכות. עליו גם להציג בפירוט כיצד יש ליישם את המדיניות בפעילויות היומיומיות של הארגון.

הגדרת מדיניות איכות טובה והפגנת מחויבות ההנהלה כלפיה היא הדרך הטובה ביותר להשגת מחויבות אמיתית של העובדים, שעליה בנויות רוב מערכות האיכות המצליחות.

כיצד צריכה להיראות מדיניות איכות טובה? אין כללים חד-משמעיים לכך, העיקר הוא שמדיניות האיכות שלך תשקף את מטרותיך, את יעדיך ואת תרבות העסקים שלך. עם זאת, יש מספר קווים משותפים לכל אלה.

שש כותרות למדיניות האיכות

1. **הצהר על כוונתך לפעול על פי התקנים**

   (אם אמנם זוהי כוונתך).

2. **הצהר על מחויבות ללקוחותיך**

אמירות כגון 'אפס תקלות', או 'נכון תמיד כבר בפעם הראשונה', אין בהן תועלת רבה, אלא אם תסביר למה בדיוק אתה מתכוון בהן. הכרזות בדבר רמת הבדיקה שאתה מבצע, או על תגובתך לתקלות, יהיו בוודאי בעלות משמעות רבה יותר.

3. **הכרז על מידת האחריות של עובדיך לאיכות**

הודעה ברורה שכולם אחראים לאיכות, ושיישפטו על פי מידת המחויבות שתופגן על ידם.

4. **הכרז על מחויבות ההנהלה שלך לאיכות**

במיוחד, עליך להבהיר שההנהלה תתמוך בפונקציית האיכות ותימנע תמיד מלאכוף שיקולים מסחריים על שיקולי איכות.

5. **זהה מספר יעדי איכות ברורים**

אלה צריכים להיות כללים המכוונים לשיפור השירות ללקוח, כגון צמצום מספר הבעיות המדווחות מהשדה; וכן שיפור הביצועים הפנימיים, כמו צמצום עלות העיבודים החוזרים.

6. **זהה מדידות שיאפשרו השוואת ביצועים מול יעדים**

התחייב לפרסם מדידות אלו.

מדיניות איכות טובה מטפלת בבהירות בכל אחד מיעדי האיכות ומסייעת לזהות את הנדרש להשגתם. אם היעד הוא לצמצם את מספר העיבודים החוזרים, תוכל המדיניות לכלול דרישה למדוד את כל התקלות, כדי שאפשר יהיה לזהות את העיבודים החוזרים ולחשב את עלותם. ייתכן אף שהדבר יחייב פיתוח גישה מובנית לאימות ולבדיקת התקפות, שתוכל לאתר את מירב השגיאות בהקדם האפשרי. למדיניות ממין זה משמעויות חשובות במונחים של הדרך בה נתכנן ונקצה משאבים לפרויקטים, כדי שאלה יפעלו ככוח מניע לפיתוח מחזורי חיים ומתודולוגיות מתאימות.

מדיניות איכות טובה קובעת תמיד במפורש את תחומי האחריות. "משימתך כעובד היא להגיע לתוצאות נכונות; השגת האיכות מותנית בביצועיך. אם אינך יכול לבצע את עבודתך כהלכה, עליך להביא זאת לידיעת מנהלך הישיר. אנו נתחייב לתת לך הדרכה מתאימה ונתייחס ברצינות להצעותיך לשיפורים". זו הצהרה פתוחה ומעורפלת במקצת, אך עדיפה על מדיניות האומרת בפשטות "נכון תמיד כבר בפעם הראשונה".

תוכל לדעת אם אמנם הצלחת להציג את המדיניות הנכונה, רק כאשר תוכל לזהות רכיב של המדיניות ואת המדידה שתאשר (או לא תאשר) שהושג כל אחד מהיעדים.

ארגון ותחומי אחריות

אחת המשמעויות של מדיניות האיכות היא ארגון המשאבים בדרך שתאפשר את השגת יעדי האיכות. דבר זה אין פירושו בהכרח שיש צורך לזהות מחלקת איכות; וגם אין לראות ביצירת מחלקת איכות גישה נבונה.

הארגון עשוי להזדקק לאנשים בכל רמה, שלהם **אחריות מיוחדת לאיכות**. אין הם חייבים לעסוק בפעילויות האיכות במשרה מלאה, ואין הם חייבים להיות מומחים לאיכות. מטרת זיהוי תחומי אחריות לאיכות היא לוודא שהחלטות עקרוניות לא יתיפולנה בין הכסאותי של הארגון. במציאות, כל אחד בארגון אמור לדעת מה מצפים ממנו וכיצד יוערכו ביצועיו. על ידי הגדרת תחומי אחריות נוכל לוודא למשל, שהוגדרו ממשקים בין אזורים פונקציונליים, שהמוצרים המוגמרים אומתו, תקפותם נבדקה, וכי שאילתות הלקוחות מטופלות במהירות.

מהדרג העליון ומטה צריכה להיות אפשרות לוודא, שתחומי האחריות עבור כל האלמנטים של מדיניות האיכות מוצאים את ביטויים בהגדרות תפקידים, או בכתבי מינוי. צריכה להיות גם יכולת לבדוק, מטה כלפי מעלה, שכל תהליכי המפתח והממשקים שהוגדרו בנהלי האיכות מאוישים באנשים שקיבלו הכשרה מתאימה ומבינים את דרישות תפקידם. ולבסוף, צריכה להיות אפשרות לוודא שנבנתה לתוך הארגון שלנו יכולת להשיג את יעדי כל פרויקט, וכי ברשותנו מספר מספיק של עובדים מיומנים ומנוסים לצורך השלמתו המוצלחת של הפרויקט.

נציג ההנהלה

האלמנט היחיד בארגון המוכתב על ידי המערכת הוא מינויו של 'נציג ההנהלה' שיהא אחראי על הביצוע הכולל של פונקציית האיכות.

נציג ההנהלה אינו חייב להיות בעל מקצוע בתחום האיכות ואינו חייב לשאת במינוי הנקרא 'מנהל האיכות'. החשוב הוא שלנציג הממונה תהיה **סמכות מספקת** לצורך ביצוע התפקיד 'מר איכות' בארגון. בדרך כלל, מדיניות איכות חזקה תחייב 'נציג הנהלה' בעל אישיות חזקה כדי לוודא סילוק מהיר של מכשולים המפריעים לאפקטיביות המערכת. מינוי אישיות בכירה בעלת סמכויות ביצוע, הוא אחת הדרכים להעברת מסר ברור לעובדים וללקוחות החברה, על מחויבות ההנהלה.

הדרכה

הדרכה היא רכיב חיוני ביותר כדי להגיע לערך ופוטנציאל מקסימליים של הארגון, וביכולת להתמודד עם השינויים והפיתוחים המתרחשים בשוק. ברוב המקרים ואף למעלה מזה, עלות ההדרכה משתלמת ומתבטאת באפקטיביות העובדים וביכולתם להבין כיצד לבצע את משימותיהם.

תקן ISO 9000-3 קובע עיקרון כללי האומר שעובדים חייבים להיות מיומנים בביצוע המשימות המוקצות להם, והוא מצביע על מספר תחומי מפתח שיש לבחון, כגון כלים ייחודיים, טכניקות, מתודולוגיות, משאבי מחשב והיכרות עם היישום. התקן מזהה אף את הצורך לקיים רישומים מתאימים בדבר ההדרכה ו/או הניסיון המעשי.

יסוד הדברים הוא תהליך הרקע שנועד לוודא זמינות מספקת של עובדים בעלי הכשרה מתאימה, שיכולים לבצע את כל היעדים המתוכננים של הארגון. כדי להשיג זאת עלינו לכלול את תכנון ההדרכה כחלק משולב בקביעת יעדי האיכות. לכן, נצטרך לבצע את ההדרכה הנדרשת בצורה מבוקרת, כדי שההתקדמות תיעשה תוך בקרה ותהא תואמת לתוכניות האחרות של הארגון.

נוסף לשיקולים האסטרטגיים של ההדרכה, קיים גם צורך לזהות את ההדרכה שתידרש עבור פרויקט מסוים במקרה שזו לא נכללה כבר בתוכנית ההדרכה הכוללת. הזמן הנכון ביותר לעסוק בכך הוא שלב בדיקת החוזה, ובאחריות מנהל הפרויקט ליישמו. עם זאת, יש צורך בקישור לאחור אל תוכנית ההדרכה, כדי לוודא שההנהלה תכלול בשיקוליה ובתכנון צורכי ההדרכה בעתיד גם את ההדרכה הדרושה לביצוע הפרויקט.

אם כן, להדרכה שלוש רמות יישום. ברמת הפרט, ההדרכה ניתנת ונרשמת ברישומים של כל פרט. ברמת המחלקה, ניתנת הרשאה לביצוע כחלק מתוכנית ההדרכה. ברמה המפעלית, נקבעות הדרישות ומדיניות ההדרכה ונסקרות תקופתית; במקביל מתנהל מעקב אחר הביצוע מול התוכנית.

אחד המנגנונים לקביעת דרישות ההדרכה הוא טבלת קישורים המזהה את הכישורים הנדרשים לכל משרה בארגון. מחויבות לאספקת ההדרכה שהוגדרה בטבלת הכישורים יכולה לשמש בפועל כאחד האלמנטים של מדיניות האיכות; יתרונה הוא בהיות המחויבות הגלויה לעין עובדי הארגון בצורה שתביא להם תועלת כפרטים בארגון, ותשפר את הפוטנציאל של כל הארגון.

אחת הדרכים שיכולה לספק להנהלה שיטה מובנית לניטור ביצוע ההדרכה גם ברמת הפרט וגם ברמת הארגון היא הכללת ההדרכה בהערכה השנתית של כל עובד. בעת ראיון ההערכה ניתן לאשר לכל עובד את תוכנית ההדרכה שלו לשנה הבאה. חשוב לציין שההדרכה יכולה להינתן בכל רמה החל מהדרכה בפיקוח תוך כדי עבודה דרך קורסים חיצוניים. אפשר גם לשקול התנסות מעשית מתאימה. מידת ההתאמה של ההדרכה, או של הניסיון המעשי הניתן, היא הקובעת. יש לזכור גם את הצורך לנהל רישום של ההדרכה, מכל סוג שהוא, כדי שניתן יהיה להציג זה מול זה, את מה שנעשה ואת דרישות ההדרכה.

כחלק מהמסגרת שפורטה לעיל, יישלחו אנשים לקורסי הדרכה מסוג זה או אחר, כדי שבעתיד יוכלו למלא יעדי ביצוע מסוימים. התועלת המופקת מקורסים אלה תהיה גדולה יותר, אם האנשים המעורבים ידעו מראש מה הם יעדי הביצוע, כדי שיוכלו להתמקד ברכישת הכישורים הדרושים ולהעריך את הקורס על פי יכולתו לספק זאת.

כאשר חוזר עובד מהדרכה, מדידה כלשהי של אפקטיביות ההדרכה תסייע לוודא שהושגו יעדי ההדרכה, ותספק לחניך עצמו הזדמנות לסקור את החומר זמן קצר לאחר השלמת ההדרכה. משתי הבחינות, דוח קצר על רעיונות המפתח של הקורס יכול להיות לתועלת ולשמש כביקורת מתמשכת על האיכות ומידת ההתאמה של ההדרכה שניתנה.

לא פחות חשוב, לערוך בהקדם האפשרי (לאחר קבלת ההדרכה) תוכנית לשימוש בכישורים שנרכשו. ערך ההדרכה יאבד במהירה אם לא ימומשו הכישורים שנרכשו. תכנון המינויים לתפקידים שלאחר הקורס, חייב להיעשות כבר תוך כדי מהלך הקורס.

מבדקי איכות פנימיים

תפקיד **מבדקי איכות פנימיים** (internal quality audits) לספק סקירה בלתי תלויה על ביצועי מערכת האיכות. התהליך הוא פשוט: נדרש לכך מבקר מאומן שיסקור את העבודה המתבצעת בתחום מסוים של הארגון למול דרישות מערכת האיכות באותו תחום. הפלט ממבדקים אלו יכול להיות רב ערך. הוא יכול לכלול:

- עדויות לחולשות תוכנית האיכות המתועדת;
- עדויות לחולשות ביישום תוכנית האיכות המתועדת;
- אירועים מסוימים של אי-ציות לתוכנית האיכות והצעות לפעולה מתקנת;

- זיהוי בעיות מסוימות שיש צורך לטפל בהן באמצעות המנגנון לפעולה מתקנת ;
- זיהוי נהגים טובים שיש מקום לעודדם במקומות אחרים בארגון ;
- נתונים על ביצועי מערכת האיכות.

על ידי תכנון מראש של **מבדקי האיכות** (quality audits), הארגון יכול להבטיח ביקורת בכל התחומים על בסיס שגרתי, וכי כל הנושאים הקשורים לאיכות יבוקרו על פי לוח זמנים סביר. תוכנית המבדקים הפנימית היא האמצעי הראשי של הארגון שבאמצעותו הוא יכול לבדוק ולהיווכח שהמערכת פועלת ביעילות כל הזמן. מטרת המבדקים היא לוודא שכל תחום ותחום שבארגון יבוקר באורח סדיר ושהתחומים היותר חשובים או קריטיים יבוקרו בשכיחות התואמת את חשיבותם או את הקריטיות שלהם.

הבסיס לכל המנגנון הוא תוכנית המבדקים הממפה את הביקורות שתיערכנה על פני תקופה מסוימת, שנה בדרך כלל. תוכנית ביקורת בנויה כהלכה ממחישה את עדיפויות הארגון ומספקת תשתית מוצקה להערכת הביצועים. ככל שהתוכנית הולכת ונפרשת, יכול המצב להשתנות ולחייב תכנון חוזר. בדיקות שגרתיות חוזרות של התוכנית יכולות לוודא שהיא ממשיכה לשקף את החשיבות וקריטיות התחומים שנבדקו. תרשים 5.3 הוא דוגמה לדרך רישום טובה עבור תוכנית לעריכת מבדקים, המראה את מצבה של כל פעילות ביקורת.

Key:
Scheduled
Completed
Actions complete
Actions verified

Area \ Date	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Software development	Actions verified					Scheduled						Scheduled
Computer operations		Actions complete							Scheduled			
Purchasing			Actions verified									
Training				Actions complete								
Installation design					Completed							
Hardware development							Scheduled					
Production control										Scheduled		

תרשים 5.3 דוגמה של תוכנית מבדק פנימית

תוכנית המבדק הפנימית מחוללת מידע ספציפי רלוונטי לתיקון בעיות ולחיפוש אחר שיפורים, אך אפשר להשתמש בה גם לבניית תמונה של ביצועים כוללים של מערכת האיכות. זו יכולה לשמש כקלט חשוב לסקירה התקופתית של מערכת האיכות על ידי ההנהלה הבכירה.

מובן שיש לכך גם מגבלות. הביקורת פוגמת בפעילויות היצרניות וצורכת משאבים שנדרשים כדי לערוך את המבדק ולדווח על תוצאותיו. קיימת גם סכנה שבודקים פנימיים קפדניים ונלהבים מדי לתפקידם, עלולים לגרום לחיכוכים בתוך הארגון ולצמצם את מסירות העובדים לנושא האיכות. יש לחשוש גם להתעלמות מנושא האיכות בפרק הזמן שבין המבדקים, מתוך ציפייה שהביקורת תתקן כבר את כל הליקויים. הבעיות האלו יכולות להתעורר וגם תתעוררנה בארגון, אלא אם תרבות האיכות תהיה מושרשת די הצורך, ובחירת המבקרים תיעשה בזהירות ותתבסס על התאמתם לתפקידם.

המבקרים הפנימיים יזדקקו להדרכה כדי לוודא שתקני של הביקורת יהיו הולמים ועקביים. כדי לוודא שהמבקרים יהיו מודעים למדיניות האיכות הכוללת של הארגון וליעדי הביצוע השוטפים יש צורך במידה מסוימת של הדרכה פנימית, או מפגשים עם נציג ההנהלה. המבקרים יכולים להוסיף ערך בכך שיזהו מצבים בהם הנהגים אינם תואמים למערכת האיכות המתועדת, ועל ידי דיונים עם התחומים המבוקרים על מידת האפקטיביות של המערכת הקיימת. צריך לקיים מרוות פעולה שיעודד הגירת נהגים טובים מתחום אחד לתחומים אחרים של הארגון, ושיזהה היכן תחום מסוים נוטה לפגר (או להשתער קדימה) לעומת התחומים האחרים, ובכך יסייע לכלל ביצועי הארגון להיצמד למטרה. כתב מינוי ברור ומפורט עבור כל פעולת ביקורת, בלוויית רשימות תיוג שתורכבנה עבור כל מבדק ספציפי, יסייעו למבקר להיות ממוקד די הצורך. מעקב אחר הדוחות יספק הזדמנויות לתשאל את המבקרים, ולהשוות לטווח הארוך את ביצועי המבקרים, מתוך מגמה להשיג עקביות בנהגי וכללי הביקורת.

אחד החלקים החשובים יותר בתפקידי נציג ההנהלה הוא **פיקוח על ביצוע תוכנית המבדק הפנימי**. יש חשיבות מיוחדת לבדיקת פעולות המעקב ולזיהוי הצורך בהסלמה עם ההנהלה במקרים בהם נראה שהפעולה המתקנת נמשכת זמן רב מזה שסוכם עליו מלכתחילה, או ממה שנחשב כזמן הולם בעיני המבקר. אלה מהווים חלק חשוב של 'כיוונון' המערכת. בארגונים רבים משתמשים במדידות פשוטות כדי לזהות היבטים חלשים יותר של המערכת (כלומר, סעיפי התקן בהם מדווחת כמות גבוהה ביותר של אי-ציות לתקן), ואת מצב הפעולה המתקנת שננקטת בכל תחום של הארגון. מידע מסוג זה יכול לסייע בזיהוי מוקדם בעייתיים לפני שאלה הופכים לבעיות חמורות.

פעולה מתקנת

המנגנונים לפעולה מתקנת כוללים נהלים שמטרתם לזהות בעיות, או בעיות פוטנציאליות, במוצרים או בתהליכים. המידע שמתקבל מאפשר לשפר את הנהלים או למנוע אירועים עתידיים של הבעיה.

בתחומי התחזוקה והשיפורים משתקפת מחויבות ההנהלה בצורה הבולטת ביותר.

יש צורות שונות למנגנוני הדיווח על פגמים, החל בתלונות לקוחות, דרך שאילתות למרכז התמיכה, פגמים המתגלים בעת בדיקה, ועד לשגיאות הנחשפות בסקרים. כל דבר המשקף את מצב איכות המוצרים, או התהליכים עשוי להיות כלי דיווח על פגמים מסוג כלשהו.

בכל מקרה יש צורך באמצעים שישמשו לתיקון הפגם, ויומן תקלות יוכל להיות שימושי ביותר, כדי לוודא שבכל אחת מהתקלות הטיפול 'נסגר'. זהו קו ההגנה הראשון. המערכת תוכל להיות יעילה יותר אם תנסה לזהות את מקור הפגם ולחסל אותו, על ידי כך שתפנה את הפעולה המתקנת לתהליך היסוד וגם לפגם עצמו. זהו קו ההגנה השני.

את התועלת הגדולה ביותר ניתן, כמובן, להפיק מכך שנתבונן בכל מנגנוני הדיווח השונים כדי לזהות סיבות משותפות ותחומי חולשה. תוכנית המבדק הפנימי תוכל לעקוב אחר הסיבות, כדי לספק בחינה מעמיקה יותר של התהליכים הנתונים בסימן שאלה.

יש גם מקום להגדיר מדידות מסוימות של השיפורים, כדי שישמשו לבקרת הביצוע העתידי של השיפורים בתחומי היעד. גישה זו מציעה לא רק את היכולת לקיים את ביצועי מערכת האיכות אלא אף לשפרם. זהו הבסיס לשיפורים מתמידים.

עשרה צעדים בדרך לפעולה מתקנת יעילה

1. **זהה בעיות**

השתמש בדוחות על פגמים, תלונות לקוחות, מבדקים פנימיים, או כל דבר החושף בעיות.

2. **חקור את הבעיות**

חשוף את אופי הבעיה ונסה לזהות את הסיבות שגרמו לה.

3. **תקן את הבעיה**

טפל בבעיה המיידית כדי שתוכל לסלק את הסימפטומים של הפגם שדווח עליו.

4. **חפש מקרים אחרים**

ייתכן שהבעיה היא אירוע יחיד, הראשון מתוך רבים, או שזוהי בעיה שכיחה אופיינית שלא נחשפה קודם לכן בצורה זאת.

5. **הבא בחשבון את המשמעויות הרחבות יותר**

ייתכן שהבעיה אירעה במקום אחר בצורה שונה במקצת, או שתחומים אחרים יכולים להיות חשופים לה, אם כי עד עתה לא אירעה בהם תקלה כזו.

6. **זהה את הסיבה השורשית**

נסה לגלות מדוע אירעה התקלה, על ידי כך שתתבונן בכל הממצאים ותנסה לזהות את הבעיה היסודית, ולא רק את הנקודה בה התגלתה הבעיה לראשונה. השתדל לתקן את הבעיה האמיתית ולא רק את הסימפטומים.

7. **אמוד את מידת הסיכון של חזרות על האירוע**

לפני שתחליט על פעולה מתקנת, נסה לאמוד אם יהיה בה כדי למנוע או לצמצם משמעותית את הסיכון של מופע חוזר של התקלה, או הבעיה.

8. **סלק את הבעיה השורשית**

נקוט בפעולה מתקנת כדי לסלק את הסיבה השורשית של הבעיה ונסה למנוע את חזרתה. דבר זה יכול לחייב נקיטת סדרת פעולות ולא רק פתרון פשוט אחד.

9. **עדכן את תוכנית המבדקים הפנימיים**

לעקוב בקפדנות אחר הפעולה המתקנת כדי לוודא את האפקטיביות שלה. דבר זה יחייב מבדקים פנימיים שכיחים יותר וייתכן שיהיה גם מקום להגדיר מדידות נוספות למעקב אחר הביצועים.

10. **אמוד את חומרת הבעיה**

אם הבעיה חמורה או נפוצה במיוחד, או אם אתה צופה שהיא תהפוך לבעיה גדולה יותר בעתיד אם לא תינקט פעולה בדרג בכיר יותר, הוסף את הבעיה לסדר היום של פגישת הסקר התקופתית, או המיוחדת, של ההנהלה.

סקר חוזר של ההנהלה

סקר הנהלה (management review) היא בדיקה שיטתית של ביצועי מערכת האיכות, המתבצעת על ידי מנהלים בכירים. מטרתו לוודא שמערכת האיכות ממשיכה להיות אפקטיבית, על ידי טיפול בסוגיות או בבעיות שמחייבות פעולה מצד ההנהלה.

תקן ISO 9000-3 ממליץ שסקר ההנהלה יכלול בדרך כלל הערכה של תוצאות ביקורות האיכות הפנימיות, שיכולות לשמש כבסיס. עם זאת, יש ביכולתנו להתבונן עמוק יותר במערכת, רק על ידי סגירת הלולאה וגם על ידי בדיקת המצב והביצועים של מנגנוני הפעולה המתקנת. מבט בתלונות הלקוחות ובתגובות הספקים יכול לספק מבט לעומק, אל הביצועים החיצוניים של המערכת. במקרה שהגדרנו מדדים למוצרים ולתהליכים (ראה פרק 6) שנועדו לאפיין את ביצועי המערכת, ניתן לבודקם מול יעדי הביצוע כדי לזהות תחומים חלשים.

תמצית של פגישת סקר הנהלה אפקטיבי מבוססת על הגדרת מינימום המשתתפים בפגישה ועל נושאי חובה החייבים להיכלל בסדר היום. נציג ההנהלה הוא זה שצריך להיות אחראי לבניית דוח שיעסוק בכל הסעיפים שעל סדר היום ויספק את הבסיס לדיונים בבעיות מפתח. הוא צריך גם להיות אחראי לרישום הדיונים ולנהל מעקב אחר הפעולות הנדרשות.

זהו תחום חשוב ביותר וצריך להתייחס אליו ברצינות. ההנהלה אמורה להיות ההנהלה הבכירה ולכלול מספר גדול ככל האפשר של חברי הצוות הביצועי. נושאי סדר היום חייבים להיות יסודיים, צופים אל העתיד ובוחנים את העבר. עבור ההנהלה זוהי הזדמנות לוודא שמדיניותה מתבצעת בהצלחה, ושהוקצו המשאבים והארגון הדרושים להשגת מדיניות זו.

תשומת לב בלתי הולמת לסקר זה היא סימפטום קלאסי למחויבות רפויה מצד ההנהלה ומוסיפה על הבעיות שנגרמות על ידי מדיניות שאינה מוגדרת כל צרכה.

מדידות ומדדים

המדידות והמדדים (metrics and criteria) נזכרים כאן כדי לבסס את מעמדם הראוי בתור אלמנט יסוד במנגנון שיפור התהליך. ניתנו כאן מספר דוגמאות, אך הרחבה רבה יותר ניתנת בפרק 6.

יצירת מדיניות מדידה אפקטיבית היא פרויקט לטווח ארוך המחייב את כל תשומת הלב שיכול לתבוע פרויקט גדול, אך אין בכך כדי למנוע בדיקה מוקדמת במספר מדדים מוגבל, שנועדה להשיג אך ורק שיפורי תהליך צנועים. זהו סוג פעילות בסיכון נמוך, שיכול לשמש כמדידת חלוץ אידיאלית, ושאפשר להשתמש בו לרכישת ביטחון בשימוש במדידה כבכלי ניהולי.

בקרת התיעוד

בקרת התיעוד המשפיע על האיכות, היא דרישה בולטת שמיושמת בצורה גרועה בארגונים רבים. תקן ISO 9000-3 מסייע לפחות בזיהוי המסמכים שטעונים בקרה: כל אלה שמגדירים נהלי איכות; כל המידע הנוגע לתכנון ולהתקדמות; כל תיעודי המוצרים, כולל המוצרים המוגמרים של הפאזות של מחזור חיים.

כלי הבקרה החיוניים גם כן ברורים: כל המסמכים שנתונים לבקרה חייבים לקבל אישור לפני הפקתם; כל השינויים במסמכים שנתונים לבקרה צריכים להיות מאושרים על ידי הרשות המנפיקה; כל הסוגיות הרלוונטיות שבכל המסמכים צריכות להיות זמינות במקומות בהם יזדקקו להן.

יש לציין שבקרת תיעוד יכולה להיות מאוד ביורוקרטית, ואין ספק שיש מקרים בהם תהליך האישור מונע את ההנפקה המהירה של מוצר מוגמר הנדרש בדחיפות. לעומת זאת, **יותר בעיות נוצרות על ידי בקרה גרועה של תיעוד מאשר על ידי בקרת יתר.**

רשומות האיכות

הרשומות נדרשות כהוכחה שאכן בוצעו פעילויות האיכות. אין בכך משמעות מיוחדת עבור מי שעוסק בפיתוח התוכנה - פרט לאותו צורך לנהל רשומות מתאימות לשם תמיכה במערכת האיכות: **רשומות איכות** (quality records).

רכש

בדרך כלל, רכש הוא נושא בעל חשיבות מועטה בענף התוכנה. לגבי מפתחי התוכנה הרכש החשוב ביותר הוא רכש כלים; הבחירה והיישום של כלים אלה מטופלים ביתר הרחבה בפרק 4. נושאים משמעותיים אחרים יכולים להיות הפעלה של קבלני משנה בעלי התמחות בנושאים ייחודיים, ובעיקר בקבלנים להדרכה.

תחום ראשי הראוי לתשומת לב יהיה לרוב ההערכה והבחירה בקבלני משנה. כללי היסוד פשוטים למדי: החלט על מה שאתה מעונין בו; שקול את המדדים החשובים

ביותר לגבי ספקים פוטנציאליים ותעד אותם; הערך את הספקים הפוטנציאליים על פי אותם מדדים ובחר על פיהם. השוני היחיד המאפיין את קבלני המשנה לתוכנה, יכול להתבטא בכך, שהיחסים ספק/לקוח, משנקבעו, יכולים להישרד משך זמן רב. האפשרות לעבור לספק חדש במקרה שהעניינים אינם מסתדרים יכולה להיות לא מעשית במקרה של ספקים בעלי התמחויות מיוחדות. לכן, חשוב לנסות להקים מערכת יחסים שמבוססת על טווח ארוך, בה שני הצדדים שואפים לשיפורים מתמידים. עליך להמשיך ולעקוב אחר ביצועי הספק שלך. בעיות פוטנציאליות יטופלו על ידי משא ומתן ותמיכה הדדית ולא על ידי הפעלת סעיפים עוינים בחוזה.

שיפור התהליך

עתה צירפנו יחד את שלושת מרכיבי המפתח של מערכת אפקטיבית לניהול איכות התוכנה:

- מדיניות ונהלים ל**ניהול פרויקטים** (פרק 3);
- מדיניות ונהלים ל**פיתוח תוכנה** (פרק 4);
- מדיניות ונהלים עבור **לב מערכת האיכות** (פרק 5).

כל מערכת איכות התואמת מבנה זה ומטפלת בסוגיות שנידונו בצורה מציאותית, תהיה מסוגלת לעמוד בדרישות התקנים ISO 9001 ו- ISO 9000-3. חשוב יותר, תהיה זו מערכת אפקטיבית באמת.

עד עתה התמקד הדיון בעיקר בכך שמערכת האיכות תשמור על קיום דרישות תקן ISO 9001. דבר זה הוא בסדר כשלעצמו, אך אפשר להפיק ממנו תועלת רבה יותר. אנו יכולים לכוון את עיקר הפעילויות שלנו להשגת שיפורים של ממש, בכך שנהיה יותר שאפתנים בציפיותינו ממדיניות האיכות ובהקצאת המשאבים שנזדקק להם, כדי להשיג מטרות איכות נמרצות יותר.

שיפור תהליכים אינו שונה ביסודו מתחזוקת מערכת איכות, אלא ששני תחומים טעונים הרחבה משמעותית:

- נוסף על תיקון חולשות, מנגנוני הפעולה המתקנת חייבים להיות מסוגלים להשיג שיפורים בפועל.
- המדידות חייבות להיות מתוחכמות יותר, כדי לאפשר את מדידת הביצועים הכוללים של מערכת האיכות.

רכיב המדידות הוא החשוב והקשה ביותר בשיפורים אלה. דבר זה יידון בפירוט רב בפרק 6. סוגיות אחרות יכללו את אימוץ מחזורי חיים וטכניקות חדשות. אלה יידונו בפרקים 7, 8 ו- 9.

עד עתה הגענו רחוק ככל האפשר בעמידה פשוטה בתנאי תקן חיצוני. מכאן ואילך הדברים תלויים אך ורק בנו.

עשרה צעדים - מתחזוקה לשיפור מתמשך

1. עצב מודלים לכל תהליכי המפתח, כדי שתכיר את כל נקודות הבקרה והממשקים.
2. זהה בעיות ותן להן קדימויות, כדי שתוכל לזהות איזו מהן ראויה להיות ראשונה לטיפול.
3. בחר בחמש הבעיות הראשונות כמועמדות התחלתיות לשיפור תהליכים.
4. תכנן אסטרטגיית שיפורים שתקיף את חמשת התחומים שבחרת.
5. הגדר יעדים מכומתים לשיפורי תהליכים, כדי שתוכל לקבוע אם היתה לשינוי השפעה מועילה על המערכת בכללותה.
6. יישם את השינוי הראשון על ידי הפעלה מלאה של תוכניתך בתחום אחד.
7. עמוד על משמר אפקטיביות השינויים על ידי מדידת השפעתם על המערכת בכללותה. במקרה שלא חל שינוי או שיש הידרדרות, חזור אל צעד 6 ושקול מחדש את השיפור.
8. יישם את השיפור הבא בסדר הקדימויות, לאחר שהשיפור הראשון יושם כהלכה ויתחיל לתת פרי.
9. קדם תחום נוסף מאלה שברשימתך עד לגמר הטיפול בכל התחומים הבעייתיים הנותרים.
10. המשך את התהליך ללא הגבלה.

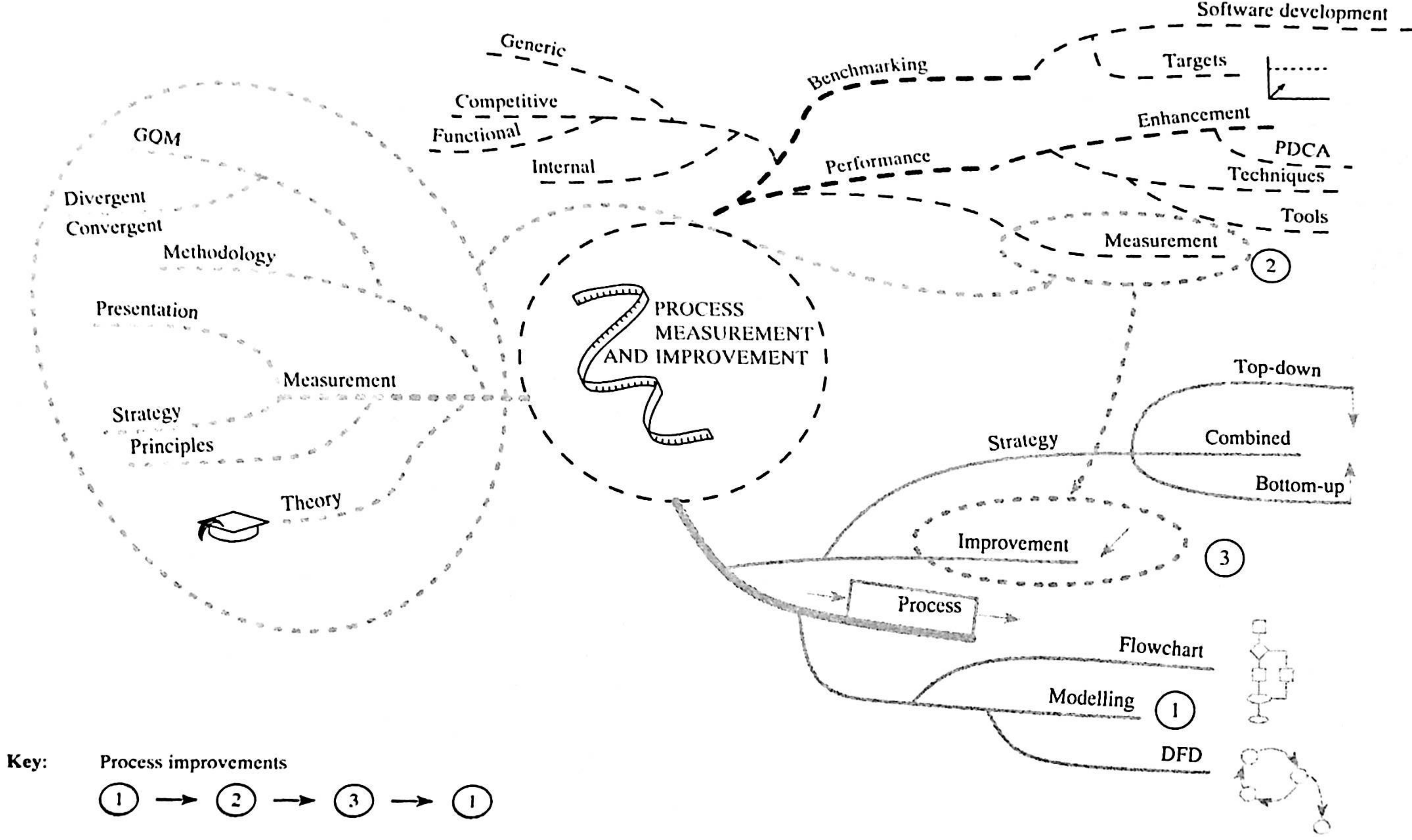
חלק 3

חלק 3 של ספר זה מורכב מפרק אחד בלבד - פרק 6: מדידה ושיפור תהליכים. מטרת פרק 6 היא להציב את המדידה בהקשרה הנכון בתוך המערכת לניהול איכות, ככלי שנועד לשקף את ביצועי העבר ואף לחזות את ביצועי העתיד. בפרק זה נבחן את הנחיות תקן ISO 9000-3 בתחום המדידה ונדון במשמעויות המעשיות שלהן.

לאחר הצגת מספר רעיונות בנושא המדידה, נעיין ברעיון ה'תהליך', אך הפעם נהרהר בדרכים לשיפור תהליכים. נדון באסטרטגיות לשיפור המשתמשות בטכניקות מדידה, כדי לקבוע מטרות לטווח ארוך שאינן עוסקות בשיפורים שרירותיים, אלא בהשגת מטרות כדאיות וברות-הישג.

חלק זה של הספר מהווה ציר מרכזי, משום שהוא מציג את המנגנון החשוב ביותר המאפשר לקיים בצורה אפקטיבית את המערכות לניהול איכות, ואף מספק אמצעים להמרת מערכות המבוססות על תקנים חיצוניים למערכות המציבות ומשיגות את מטרותיהן הפנימיות להשגת שיפורים מתמידים.

רעיונות המפתח של חלק זה מוצגים בתרשים ח.3.1, מפת התפיסה של הפרק.



תרשים ח.3.1 מפת התפיסה של חלק שלישי

פרק 6

מדידה ושיפור תהליכים

מבוא

המדידה (measurement) היא אבן יסוד בהנדסה ובמדע. כדי שיתייחסו אליה ברצינות, חייבת הנדסת התוכנה לפתח דיסציפלינת מדידה כלשהי. הבעיה שניצבה תמיד בפני מהנדסי התוכנה התבטאה בשאלה מה עלינו למדוד. המחסור החמור במודלים של תהליכי פיתוח תוכנה מקובלים על הציבור, הקשו על אפשרות השוואת קבוצה אחת של נהגים עם קבוצה אחרת, או אפילו על היכולת לשפוט אם קבוצה נתונה של נהגים יש בה כדי לספק מוצרי איכות. חייבים להיות לנו מודלים ומדידות של תהליכים אשר יאפשרו לנו להעריך, ובסופו של דבר גם לשפר, את איכות המוצר והתהליך. אין די בכימות ובשיפור תהליכים; עלינו גם להראות שהמוצרים המוגמרים עונים על דרישות מוגדרות ומכומתות.

אבני היסוד של המדידה

קצת תיאוריה

מה הם העקרונות הבסיסיים של המדידה? על פני השטח, המדידה, בדומה לרבים מענפי המתימטיקה, פשוטה למדי, עם זאת היא מסוגלת לבלבל לחלוטין את הזרים לנושא. אין בכוונתנו להעמיק ולחקור בתאוריות, אך עלינו להבין את כללי היסוד בעזרתם נוכל למנוע שימוש לא-נכון במדידות, ולהימנע מלהגיע למסקנות שגויות בעקבות מדידות אלו וה'ניתוחי' שלהן.

הבה נתחיל בהגדרה 'פשוטה':

> *מדידה - הקצאת יעד אמפירית של מספר (או סמל) לישות כלשהי, כדי לאפיין תכונה מסוימת.*
>
> *(פנטון, 1991)*

מונחים טכניים יכולים לערפל בקלות את המשמעות, אך במקרה זה נזדקק לדיוק מסוים בהגדרה מרכזית זו.

מדידה עוסקת בשימוש במספרים או בסמלים לייצוג **מאפיינים** (attributes) או **דברים (ישויות** - entities) שאנו מעוניינים בהם. לדוגמה, אנו מעוניינים במאפיין העובי של ספר (ישות), או באורך תוכנית, או בעלויות שיגרום לנו פגם בתוכנה. כשאנו אומרים שהספר הוא בעובי שישה ס״מ אנו מקצים מספר למאפיין העובי, ומספר זה יאפיין את הספר בכל הנוגע לעוביו - הוא דק יותר מספר שעוביו 12 ס״מ ועבה יותר מספר שעוביו 3 ס״מ. מובן שלא נוכל להסיק מכך דבר באשר לתוכן הספר, או לאיכותו.

הרעיון שהמספר המוקצה צריך להיות אמפירי ואובייקטיבי, אומר למעשה, שעל המספר להיות מבוסס על התבוננות ועל הניסיון המעשי, ולא להיות מותנה במבצע המדידה. כשאני אומר שעובי הספר הוא שישה ס״מ, אבל אתה מודד אותו ומוצא שעוביו תשעה ס״מ, לא צריך להיות מקום לספק באשר לאמת. כנראה שאחד מאתנו שגה במדידה, אחרת אין למדידה משמעות.

הקושי שבמדידה אמפירית ואובייקטיבית מודגם על ידי הדוגמאות שזה עתה ראינו. מהו אורך תוכנית? זו נראית שאלה ישירה מאוד, אך בפועל, המדידה מותנית בשאלה האם אנו סופרים משפטים או שורות, האם אנו כוללים הערות או לא, וכך הלאה. אין ספק שאורך תוכנית יכול להימדד בדרך אמפירית ואובייקטיבית, אלא שעלינו להיזהר בבחירת המונחים שאנו משתמשים בהם; אחרת אנו מסתכנים בחוסר בהירות במדידה שלנו, רק משום שיכולים להיות לנו רעיונות שונים באשר למשמעות הפרמטרים.

מאידך, עלות של פגם, היא הרבה יותר בעייתית. בכל הנוגע לאורך תוכנית, הרבה יותנה בדרך בה אנו מודדים. אלא שכאן ניצבת בפנינו בעיה נוספת והיא, שאנשים שונים מודדים בשל נימוקים שונים. למונחים יש סיבות שונות עבור אנשים שונים משום שהם שוקלים אותן מנקודות ראות שונות. הסיבה אינה משום שאיננו יכולים להסכים על המשמעות, אלא משום שאיננו יכולים להסכים על משמעות שהיא בעלת אותו ערך לכל אחד מאתנו.

העלות היא מסוג המדידות שיש להן משמעות שונה לאנשים שונים, על פי היחס והקשר שלהם לבעיה. לקוח המודד את עלות הפגם בתוכנה שלנו יכלול בדרך כלל רכיבים שאנו לא היינו כוללים. אנו עשויים להתעניין כמה יעלה לנו לתקן את הפגם, כולל עלויות נוספות הנגרמות כתוצאה מהפרעה בעבודה אחרת, אשר גורמת לנו לחפש את הפגם. הלקוחות לא יתעניינו בעלויות הנוספות שלנו, אלא בעלויות הנוספות שלהם, למשל, העלות של עסקים שנדחו או אבדו. הגדרת אמצעי אמפירי ואובייקטיבי למדידת עלות הפגמים, בה נוכל להשתמש באופן אוניברסלי אינה בעיה פשוטה, וייתכן שלא יימצא עבורה פתרון.

עלינו לשאוף להשגת הגדרות מקומיות למדידות כאלו, שיש לעשותן בזהירות רבה, כדי שגם האחרים ידעו איזו משמעות לייחס למדידות שלנו. הם עשויים להשתמש במדידות אחרות, או להגדיר אותן מדידות בדרך שונה, אך כולנו צריכים לדעת אם יש ביכולתנו להשוות בביטחון את המדידות השונות שלנו. טוב יותר יהיה לכולם, אם המדידות תהיינה בנות-השוואה. אסור לנו להניח מראש שקיימת אפשרות להשוואה כאשר זו אינה קיימת.

עתה, לאחר שעומדת לרשותנו הגדרה שאפשר לפעול לפיה, יש עוד להבהיר כמה רעיונות לפני שנוכל להתחיל להשתמש בה. הרעיון הראשון עוסק בנושא הייצוג. מן ההכרח הוא שאפשר יהיה לומר, שכל מספר או סמל המוקצים לתכונה כאמצעי מדידה, אכן מייצגים את אותה תכונה. כלומר, הקצאת מספרים תואמת להבנתנו את עולם המציאות - סֵפר ארוך יותר יוצר ערך מִספרי ארוך יותר לייצוג אורך הסֵפר.

הרעיון השני, ההקצאה של מספר מסוים חייבת להיות בלבדית. ייתכן ששתי מדידות שונות יקצו מספר שונה לאותו מאפיין, במקרה זה צריכות המדידות עצמן להתייחס זו אל זו בבהירות ובצורה מתאימה. לדוגמה, מדידה של אורך תוכנית הכוללת הערות, ומדידה שאינה כוללת הערות, יתנו ערכים שונים עבור אותה תוכנית, אך נוכל להמיר בקלות את הערך האחד לשני ולהיפך.

לבסוף, המדידות חייבות להיות בעלות משמעות. למעשה, זהו רעיון מורכב, אך נוכל להסתפק בהבנה פשוטה שלו, לפחות עד כמה שהדבר נוגע לתחום ההתעניינות המיידי שלנו. מדידה בעלת משמעות, היא מדידה המאפשרת לבנות עבורה משפטים 'שיש בהם היגיון', וכל התמרונים הסטטיסטיים שעלינו לבצע, חייבים להיות תקפים לגבי מדידה זו. תוכנית C בת 50 שורות, אי-אפשר לומר עליה שהיא ארוכה כפליים מתוכנית 'פסקל' בת 25 שורות, גם אם נהיה זהירים מאוד בהגדרת 'שורה'.

התמיכה התיאורטית בהגדרה של מה שמהווה את המדידה היא כמובן בעלת חשיבות, ועלינו להיזהר לא לפגוע בכללים, בנסיון לתמרן בין המדידות.

מתודולוגיית המדידה

פרט אחרון לפני שנצא לדרך המדידות: אנו זקוקים למתודולוגיה לביצוע המדידות, כשם שאנו זקוקים לה לפיתוח תוכנה - כדי שתספק לנו מסגרת המכילה קבוצת כללי עבודה שיסייעו לנו לבחור את היישויות והמאפיינים הנכונים עבור המדידה. בעזרת המתודולוגיה אנו יכולים לתכנן ולבקר את הפעילות וגם לתקשר ביחס לתהליך המדידה עצמו. כך יש ביכולתנו לשפר את תהליך המדידה.

קיים תמיד הפיתוי למדוד כל דבר, מתוך תקווה שלפחות חלק מהמדידות יביא תועלת.

לדאבוננו, יש כמה ליקויים בגישה זו. בעיה ראשונה היא, שלעולם נדמה לנו שאין ברשותנו כל המידע הדרוש לנו כדי להגיע לשיפוט מאוזן על נושא כלשהו. הסיבה לכך שהמדידות שערכנו כדי לסייע לנו להגיע לכלל החלטה אינן ממוקדות די הצורך בתחום הרלוונטי. שנית, אלה העוסקים במדידה מקבלים על פי רוב משוב מועט מהמדידות שלהם, אם בכלל, ובמרוצת הזמן הם מאבדים עניין בנושא. שלישית, המנהלים, אליהם מוזרמות כל המדידות, מוצפים בפרטים בעלי משמעות מועטה לגביהם, אם בכלל, והם מגלים שיכולתם להשתמש במידע המדידה הזה הולכת ופוחתת. השיתוק שהוא תוצאת הנִיתוח הולך ומשתלט.

אלו כמובן קריקטורות, אלא שכל הקריקטורות מבוססות על האמת, ומדגישות את ההיבטים המגוחכים והמשעשעים של המציאות.

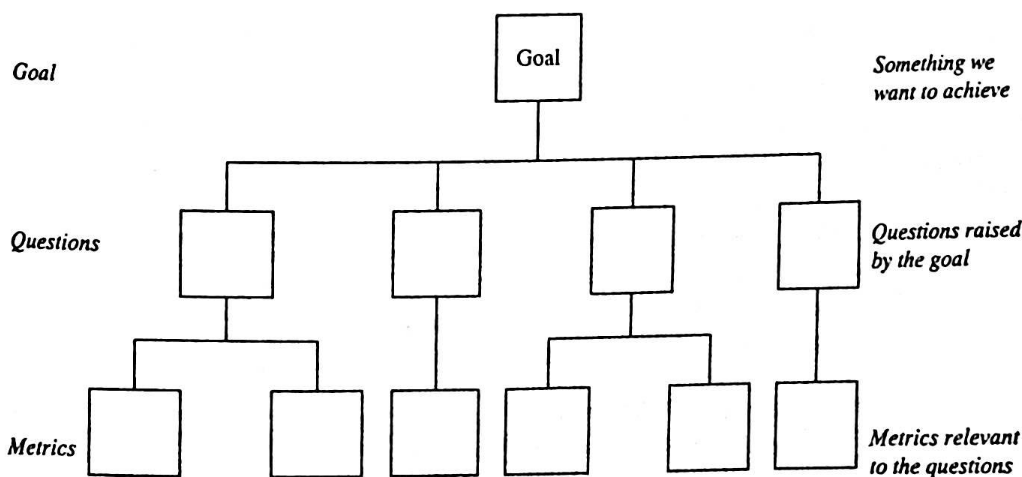
אנו נשתמש כאן בגישת 'מדד שאלת היעד' (GQM - Goal Question Metric, בזילי את רומבך, 1988) לנושא המדידה, ולו רק בשל העובדה שגישה זו הוכיחה את עצמה כהולמת ושימושית במיגוון רחב של ארגונים. זאת אינה המתודולוגיה הזמינה היחידה וגם לא בהכרח הטובה יותר, אך יש בה כדי לספק נקודת מוצא טובה.

עלינו לדעת תמיד מדוע אנחנו מודדים את מה שאנחנו מודדים.

אחת המטרות הראשיות של GQM היא לספק את המוקד הדרוש, אשר יאפשר לנו לענות לשאלות ספציפיות מסוימות, להימנע ממצבי שיתוק ומהבעיה של משוב חסר אצל העוסקים במדידה. GQM יאפשר לנו להתמקד בתחומים החשובים יותר למדידה ולנקוט בצעדים שימושיים ומשמעותיים, אשר יאפשרו לנו להגיע למסקנות ועשויות להוליך לשיפורי תהליכים. שיפורי תהליכים אינם השימוש היחיד שאפשר לעשות ב - GQM, בפרק זה נתמקד ביישום זה.

יעדים, שאלות ומדידות

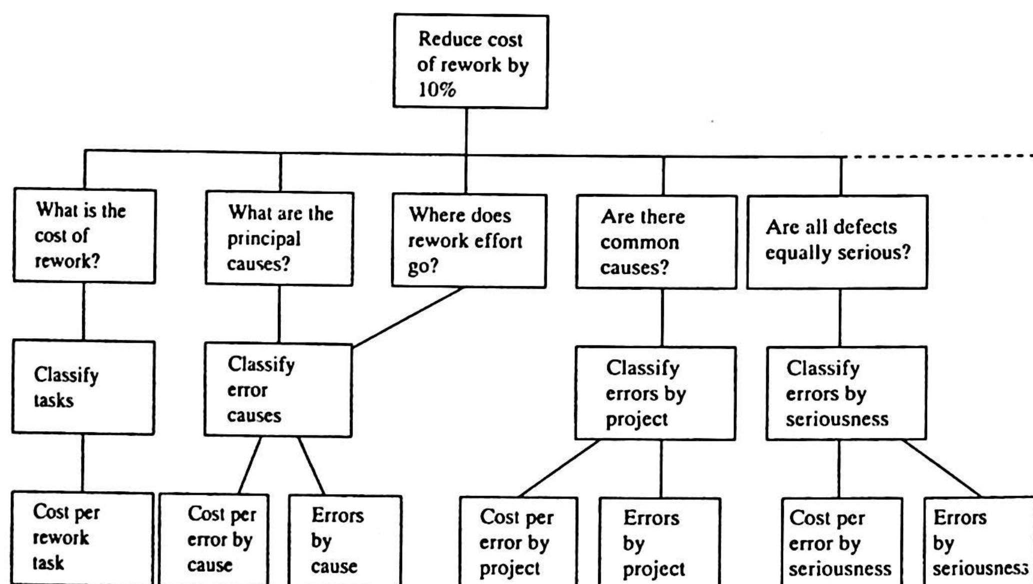
יעדים, שאלות ומדידות (goals, questions and metrics) קשורים זה לזה בקשר היררכי. יעד אחד מהווה פתח לסדרה של שאלות, ואלו מוליכות לאוסף של מדידות. מטרת כל מדידה, לסייע במתן תשובה לשאלה אחת או יותר. מטרת כל שאלה, להפיץ אור על הדברים הרלוונטיים להשגת היעד הדרוש. המודל הבסיסי מוצג בתרשים 6.1.



תרשים 6.1 *גישת GQM*

לפני שנתבונן במודל זה ביתר פירוט נשים לב שלכל מדידה (מדד metrics) תהיה כוונה ספציפית מאוד הקשורה ליעד יחיד, ולכל שאלה שאנחנו מחליטים לשאול, תוגדר מדידה אחת או יותר, כדי לוודא שכל המידע הדרוש יהיה זמין לצורך הטיפול בשאלה הנדונה. בקצרה, השגנו התמקדות ונמנענו ממדידות מבוזבזות.

מהו **יעד** (goal)? בכל הנוגע ל-GQM, יעד הוא דבר שאתה רוצה להשיג במקום שהמדידה אמורה למלא בו תפקיד. אנו יכולים לשאוף להקטין את מספר נפילות התוכנה שלנו בשדה (מסוג המטרות שעלינו להביא בחשבון, שכפי שמשתמע מתקן ISO 9000-3). נוכל לקבוע לנו יעד, כמו 'ירידה של 10 אחוז במקרי הנפילות המדווחים מהשדה'. לחילופין, אנו עשויים להיות מעוניינים לגלות אם השימוש בכלי CASE חדש ישפר את הפריון שלנו. נוכל לקבוע יעד האומר 'בדיקת ההשפעות על הפריון שתהיה לכלי XYZ של CASE'. והיעד השאפתני מכולם, אנו יכולים לשאוף לשפר את תהליך פיתוח התוכנה הכולל. במקרה זה נוכל לנסח מטרה, כגון 'צמצום עלות העבודה הנעשית מחדש ב-10 אחוזים'. בכל הדוגמאות האלו עלינו להיזהר מאוד, ולוודא בדיוק את המשמעות המעשית של מונחים כגון פריון, או עלות העיבודים החוזרים. הנקודה העיקרית היא הצורך לזהות מטרה יחידה וברורה שתהיה ברקע פעילויות המדידה שלנו.



***תרשים 6.2** ניתוח GQM של עלות העיבודים החוזרים*

מייד לאחר שנתמקד על יעד, כגון צמצום העבודה הנעשית מחדש, צצות **השאלות** (questions). האם מוצרי תוכנה מסוימים מחוללים כמות עיבודים חוזרים גדולה יותר מזו של מוצרים אחרים? האם בעיות מסוימות משותפות ליותר מאשר מוצר אחד? האם פגמים מסוימים חמורים יותר מאחרים? היכן מושקע רוב המאמץ? נוכל לסער מוחות בשאלות כאלו עד שנרגיש שזיהינו את משתני המפתח, ואז להתחיל לנסח את

השאלות החשובות ביותר לטיפול בהיבטים בעלי ההשפעה הרבה ביותר של היעד, או של הבעיה. אין לצמצום מספר השאלות חשיבות משל עצמו, אך יש בו כדי להקל על שמירת 'ניקיון' החשיבה והתמקדות בבעיה שבטיפול.

תרשים 6.2 מציג אחדים מהשלבים המוקדמים של חשיבה אודות היעד 'צמצום העיבודים החוזרים'. בשלב זה אנו יכולים לראות שעלינו להיות מסוגלים לאסוף נתונים בנוגע לשגיאות, אלא שעלינו ליצור גם מספר מרשמים עבור שיטת הסיווג, כדי שנוכל להפיק את מלוא התועלת מהמידע. סביר להניח שהמדידות שזיהינו עד עתה יסייעו לנו להבין טוב יותר את הבעיה שלפנינו, וזה עשוי להוביל לכך שנשאל מספר שאלות חדשות, או שונות. עלינו להיות מוכנים לחזור שוב ושוב על התהליך עד שתהיה לנו אסטרטגיה ברורה של הדרך להשגת היעד. היתרון של GQM בכך, שהוא מסייע לנו להתמקד בבעיה האמיתית, במקום למדוד כל דבר שנוכל רק להעלות על הדעת תוך תקוות-שווא שמשהו מכל אלה יהיה, אולי, שימושי עבורנו.

חשיבה מסתעפת-מתכנסת

חשיבה מסתעפת (divergent thinking) עניינה בקריאת דרור לדמיון ונתינת חופש ליכולת ההדמיה, בעוד אנו משאירים את כושר הביקורת שלנו מחוץ לתהליך. המטרה העיקרית היא לחולל מספר רעיונות רב ככל האפשר, מבלי לנסות להעריך אותם.

החשיבה המתכנסת (convergent thinking) נוקטת בגישה ממושמעת יותר כלפי הרעיונות שנולדו. חשיבה זו מותחת ביקורת, מסלקת כמה מהם במקרה הצורך, ומציבה את הרעיונות בסדר עדיפויות כלשהו, כדי שנוכל ליישמם בצורה מובנית.

לניגוד בין שני סגנונות החשיבה יש חשיבות רבה, משום שהוא שומר על העניין ועל ההתעסקות שלנו בבעיה שלפנינו. התעסקות רבה מדי בסגנון המתלכד (האנליטי) גורמת לנו לעייפות וללחץ - במוקדם או במאוחר אנו מתחילים להשתעמם או להירדם, ולא מתקדמים הרבה. החשיבה המסתעפת משעשעת יותר ומעוררת את היצירתיות שבנו ואת הרצון הטבעי לשחק במשחקים. בדרך כלל, אנו נהנים מסוג החשיבה המסתעפת, אך מעטים הסיכויים להפיק בדרך זו רעיונות מעשיים שניתן ליישמם.

על ידי מחזורים של חשיבה מסתעפת ואחריה חשיבה מתכנסת לסירוגין, אנו מביאים למקסימום את היצירתיות שבנו. כך אין בזבוז זמן בנסיונות ליישם 'רעיונות שנשלפים מן השרוולי', ואף נמנעים מהשעמום ומהעייפות שבחשיבה אנליטית מתמשכת.

כדאי לך לנסות - זה גם עובד וגם משעשע

בפועל, GQM עשוי להיות כרוך במספר חזרות על שלבי יעד ושאלה. היעדים עשויים לחייב הבהרה ופיצול לתת-יעדים; השאלות עשויות להזדקק לעידון ולבנייה מחדש, כדי לספק רמה הולמת ועקבית לחלוקת היעד למרכיבים. זוהי פעילות יצירתית שבעקבותיה באה בחירה קפדנית. הכוונה הכללית היא לבחור ולמסגר בזהירות את השאלות, אשר לפי הערכתנו יסייעו לנו מאוד בהשגת היעד שקבענו.

אם נציג נכונה את השאלות, או לפחות נציג אותן ברמה הנכונה, צפוי שהמדידות תבחרנה את עצמן מאליהן. מטרתנו היא לזהות מדדים שאינם חורגים מהכללים התיאורטיים ושיוכלו להיות שימושיות בקבלת תשובות לשאלות. לדוגמה, על שאלה ששאלנו, האם פגמים מסוימים חמורים יותר מהאחרים, ניתן לענות על ידי שנגדיר מספר קטגוריות לייצוג רמת החומרה של פגמים, ואחר כך נספור את מספר הפגמים ששייכים לכל קטגוריה. נוכל גם למצוא איזה מוצרים מחוללים את רוב הפגמים, על ידי שנקצה קוד לכל פגם שאנו סופרים. אם נרצה להעמיק עוד יותר בבחירה, נוכל לספור פגמים גם לפי מוצר וגם לפי מידת החומרה.

לאחר הגדרת כל המדדים, נוכל להתחיל באיסוף בפועל של נתונים לצורך מתן תשובה לשאלות, אך ניתוח מכין זה הוא חשוב מאוד ואסור להיחפז בו. **המדידה** אינה דיסציפלינה המפיקה תוצאות מיידיות, ואם לא ננהג בה בזהירות וביסודיות אנו עלולים למצוא שהיא אינה נותנת לנו שום דבר שהוא בעל ערך.

הנחיות תקן ISO 9000-3

תקן ISO 9000-3 מציע מספר הנחיות ברורות ומפורשות בדבר השימוש במדידות גם לגבי המוצרים וגם לגבי התהליכים. בתחום המוצר, ההנחיות מכירות בכך שלא קיימות מדידות המקובלות על הכל בדבר איכות התוכנה. עם זאת, ההנחיות עומדות על כך שיש הכרח לאסוף מספר מדידות מפתח, שמייצגות את המקרים המדווחים על כשל בשדה ו/או פגמים, מנקודת ראותו של הלקוח.

ההנחיות מגדירות גם את השימושים שיש לעשות במדידות המוצר:

- לאיסוף נתונים ולדיווח ערכים מדידים בצורה סדירה;
- זיהוי רמת הביצוע הנוכחית על פי כל אחד מהמדדים;
- נקיטת פעולה מתקנת במקרים של הרעה ברמות המדדיות, או של חריגה מעבר לרמות שנקבעו;
- קביעת יעדי שיפורים ספציפיים, תוך שימוש במונחים של המדדים.

מכאן משתמע שעלינו להחליט על **יעדים** (targets) עבור כל מדד של מוצר, ולהשתמש בהם לניהול תהליך פיתוח המוצר. כך נוכל לזהות אם יש נסיגה בביצוע, וגם נוכל לפעול לקראת שיפורים ספציפיים במרוצת הזמן.

בתחום התהליך, הייעוץ הוא פוזיטיבי פחות. מדדי התהליך אמורים לשקף:

- עד כמה מיטיב תהליך הפיתוח להתבצע במונחים של אבני דרך ועמידה בלוח הזמנים של יעדי איכות שבתוך התהליך.
- עד כמה מצליח תהליך הפיתוח לצמצם את אפשרות החדרת תקלות, או את אי-זיהוין של תקלות שחדרו פנימה.

אלו הן מטרות מצומצמות ביותר. בעיקרון, הן עוסקות בניהול הפרויקט ובהימנעות משגיאות. אנו מסוגלים ליותר מזה.

כיצד ליישם בפועל את רעיונות המדידה

אין לנו עניין בתורת המדידה כשלעצמה, אלא רק במקום בו נוכל לרתום אותה ביעילות ובדרך מעשית להשגת מטרות המדידה שלנו. מה שדרוש לנו הוא קבוצה של עקרונות מנחים שתסייע לנו להקים אסטרטגיית מדידה עקבית ושימושית.

נוסף לידיעת המגבלות התיאורטיות ולציות לכללים, אנו זקוקים למספר רעיונות בסיסיים ושימושיים שיענו לנו על מה שיכול, או לא יכול, לפעול בהצלחה. נוכל לקודד רעיונות שימושיים אלה לתוך קבוצה של עקרונות מדידה.

אחד עשר עקרונות בסיסיים למדידה

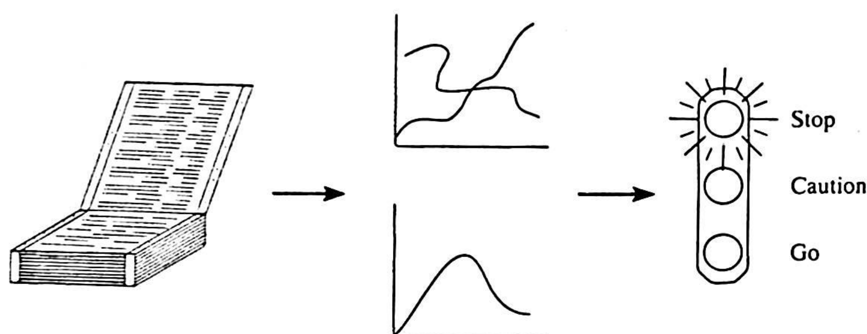
1. לעולם אל תנסה לשנות דבר עד שלא תמדוד את ביצועיו הנוכחיים ותקבע מטרות לביצועים בעתיד (אלא אם אין למעשה כל חשיבות מעשית לתוצאה).
2. לעולם אל תנסה למדוד דבר כל עוד אינך יודע לשם מה אתה מודד אותו.
3. ביחס למדידה, נקוט תמיד גישה של מלמעלה למטה כשנקודת המוצא היא מטרה או יעד-מדידה ברורים, אלא אם יש צורך בגישה של מלמטה למעלה.
4. זהה תמיד את היתרונות שעובדיך עשויים להפיק בהמשך הפעולה, לפני שתבקשם לבצע מדידות כלשהן, ודאג להזין להם את התוצאות מאוחר יותר.
5. זהה תמיד את היתרונות שמנהלך עשוי להפיק בהמשך הפעולה לפני שתבקשו לאשר את ההשקעה במדידה, אל תשכח להציג את התוצאות לכשתשיגן.
6. התחל תמיד בקטן והרחב לאט לאט. דאג להציג את התוצאות תוך כדי הרחבת הפעילות.
7. אל תצפה רק לחדשות טובות ממדידותיך, דאג שמנהלך יהיה מודע לעיקרון זה.
8. התחל בפרויקט חלוץ, פרויקט שיאשר כמה מהדעות הקדומות של ההנהלה שלך.
9. אל תצפה לתוצאות מוצלחות כבר בפעם הראשונה, ואפילו לא בפעם השנייה. אל תתלה את כל הקריירה שלך בפרויקט החלוץ של המדידה.
10. וודא שיש לצידך בכיר בהנהלה שיתמוך במאבקך.
11. אל תשתמש בעקרונות של אחרים. רשום לעצמך את קבוצת עקרונות המדידה שלך, שתהיה תואמת לתרבותך הארגונית.

הצגת המדדים

דרך הצגת המדדים יכולה להיות מכרעת לגבי קבילותם. דף פלט מהמדפסת יכול לנסוך שינה על המקבלים אותו, בעוד שתמונה גרפית ברורה יכולה לעורר שאלות נוספות, או פעולות מיידיות. חשוב ביותר לוודא שהמדדים שאתה מציג יענו על צרכיו של המקבל ויוצגו בדרך האפקטיבית ביותר.

יהיה זה רעיון טוב לערב בפעולה את היקהל' הפוטנציאלי שלך, בעוד אתה מתכנן את תוכנית המדידה ובוחר בנתוני המדידה שתאסוף. נוסף לרכישת המחויבות שלהם לתוכנית המדידה, תוכל לבקשם לזהות את המדידות שהם מצפים לראות, ואיזה שימוש הם מקווים לעשות בהן. תשובותיהם יסייעו לך להחליט מהי צורת התצוגה הטובה ביותר.

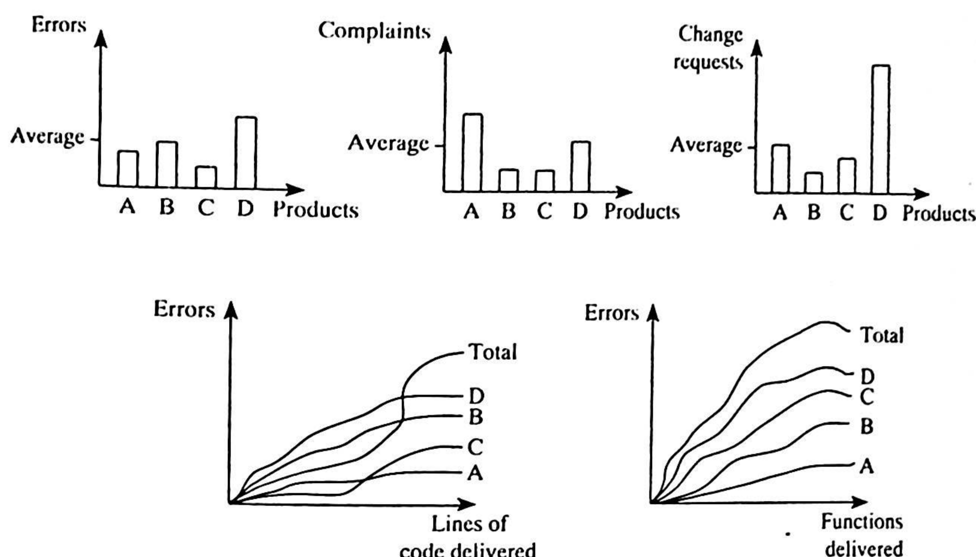
ככלל, עליך לזכור שנתונים מעובדים קלים יותר לקליטה מאשר מספרים גולמיים סתם. נתונים השוואתיים ונתונים על קצב השינויים הם לעיתים בעלי ערך רב למקבלי ההחלטות. בנוסף, זכור שמנהלים הם, בדרך כלל, אנשים עסוקים ויש להם זמן מוגבל ללימוד מדדים. חשוב שיוכלו לעכל במהירות את המידע ולראות את הקשר בין הפרמטרים. תרשים 6.3 מציג כיצד יכול עיבוד קטן של הנתונים לסייע בתהליך קבלת ההחלטות.



***תרשים 6.3* רמות עיבוד של מדדים**

דרך מקובלת להצגת מדידות מפתח היא בצורת 'לוח מחוונים'. בדומה ללוח המחוונים של המכונית, לוח המחוונים של המדדים מציג רק את המידע החשוב ביותר. בצורה זו יכולה להתגלות משמעות המידע תוך כדי סריקת הנתונים, ולא מתוך עיון ארוך ומפורט. לוח מחוונים המיועד להנהלת חברה עסקית עשוי לכלול למשל, תזרים מזומנים, רווחים, ערך הזמנות שנתקבלו, וערך עבודה בתהליך. מנהל עסוק יודע פחות או יותר למה עליו לצפות. הוא יבקש למצוא את צורת התנהגות הנתונים ובוודאי ירצה גם להשוות חודש מול חודש, או רבעון מול רבעון. רובנו משתמשים באותו עיקרון כאשר אנו בודקים את חשבון הבנק ואת חיובי כרטיס האשראי. אם הכל פחות או יותר כצפוי, איננו טורחים לרדת לפרטים.

אנו זקוקים ללוח מחוונים שונה עבור מפתחי תוכנה. מה צריך להיכלל בלוח זה? שגיאות לפי מוצר, שגיאות מול מספר שורות קוד, שגיאות לפי פונקציה, מאמץ נדרש לשורת קוד, מאמץ לפונקציה, עלות ממוצעת לשגיאה, תלונות לקוחות, או דרישות לשינויים שטרם טופלו. כל אלה מועמדים לתצוגה ויש בוודאי רבים אחרים. החיפוש אחר מה שמפתחים ירצו לראות בלוח המחוונים שלהם הוא בדרך כלל, תרגיל כדאי כשלעצמו. תרשים 6.4 מציג את עיקרון לוח המחוונים. אידיאלית, יש להציג את המדדים בצורת מונים המראים ערכים של קו הבסיס, מטרות, מצב נוכחי ואם אפשר, גם אם המדד משתנה 'בכיוון הנכון'.

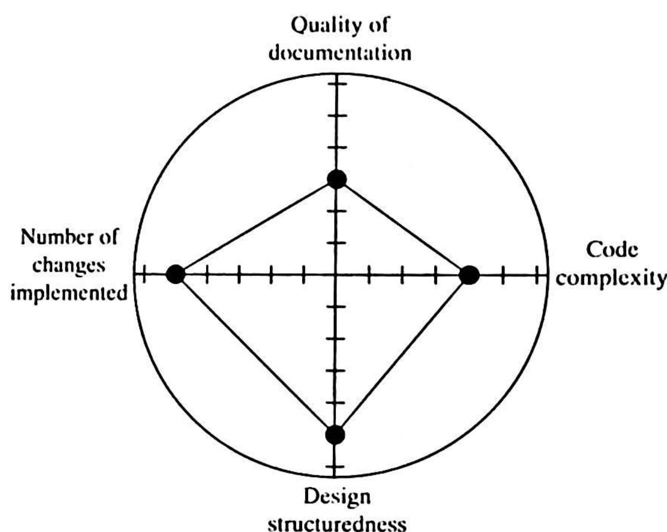


תרשים 6.4 לוח המחוונים של מפתח תוכנה

רעיון נוסף ואחרון בנוגע לתצוגה מתייחס למדדים מורכבים - מדדים המחייבים שימוש במספר מדידות להרכבת 'תמונה' של תכונה מעניינת. לדוגמה, יכולת התחזוקה עשויה להיות מושפעת מאיכות התיעוד, ממורכבות הקוד, ממידת 'התבניתיות' של התיכון, וממספר השינויים שהוכנסו במערכת. קשה לנבא כיצד ישפיעו מדדים אלה זה על זה. כל ניסיון לחזות כיצד יפעלו מדדים אלה הדדית יהיה משופע בקשיים, ולכן יש לחשוש שגם לא יזכה לאמון רב. דרושה לנו שיטה להצגת משותפת של המדדים השונים, כדי שנוכל לראות מהי ההשפעה שיש לכל אחד מהם על המדד הכולל של יכולת התחזוקה.

תרשים קיוויאט (Kiviat) הוא גרף מרובה-צירים. ראשית הצירים נמצאת במרכז התרשים, ולכל אחת מהמדידות הנפרדות יש ציר משלה שמתפשט מהמרכז החוצה. כאשר נשרטט את הערך של כל מדידה ומדידה על גבי הציר המיוחד לה, נקבל צורה אופיינית שמייצגת את המדד המורכב. תרשים 6.5 מראה כיצד יכול תרשים קיוויאט להיראות במקרה כזה.

הייצוג שבתרשים קיוויאט אינו מכיל מידע מדידתי נוקשה; הוא רק מאפשר לנו לראות את כל המדידות השונות כשהן מוצגות במשותף בתמונה אחת כוללת. כעת, נוכל לחקור את השפעת כל אחת מהאסטרטגיות המועמדות על שיפור התמונה הכוללת, כדי להגיע להבנה טובה יותר של הקשרים בין המדידות השונות, ושל סוג ההשפעה לה ניתן לצפות כתוצאה מהכנסת שינויים במדידות אלו. לאחר מספר תרגולים נוכל לקבוע 'צורות מטרה' עבור מדדים מורכבים כאלה.



***תרשים 6.5* תרשים קיוויאט עבור יכולת תחזוקה**

מה הם התהליכים

תהליך הוא כל קבוצה של פעילויות שיש להן מטרה מוגדרת וגם קלט ופלט מוגדרים. בהקשר של פיתוח תוכנה, אנו מעוניינים בעיקר בתהליך פיתוח התוכנה עצמו, כלומר בתהליך הממיר את דרישות המשתמשים למוצרי תוכנה. מחזור החיים של הפיתוח מקיף תיאור של התהליך הכולל. את מחזור החיים ניתן לפצל לשלבים, שכל אחד מהם גם הוא תהליך. ביכולתנו ללמוד את תהליך התיכון, או את תהליך הבדיקות, כל אחד בפני עצמו, וגם ללמוד את תהליך הפיתוח בכללותו.

שיפור תהליכים

שיפור תהליכים הוא כל אסטרטגיה המכוונת לגרום לתהליך שייעשה אפקטיבי יותר. לדוגמה, את מחזור החיים של פיתוח התוכנה נוכל לשפר, אם נוכל לשנותו כך שיפיק מוצרי תוכנה מהר יותר וללא ירידה כלשהי באיכותם. בפועל, נעשו כבר נסיונות ליצירת אסטרטגיות לשיפור תהליכים ממין זה. שיטות פורמליות ועיצוב מבני הם דוגמאות לאסטרטגיות שיפור עבור תהליכי פיתוח תוכנה.

שיפור תהליכים הוא שיטה מבוססת-מדידה, לשם השגת תוצאות טובות יותר.

האם הצליחה האסטרטגיה שלנו המשתמשת בשיטות פורמליות לשיפור תהליכי פיתוח תוכנה? אין ביכולתנו לומר בוודאות, משום שלא נעשה כל **מידול** (modelling) מכומת שיאפשר לנו למדוד את התהליך לפני ואחרי יישום אסטרטגיה כזו. בפועל, הרגשתינו

לגבי אסטרטגיות כאלו מבוססות על דעותינו האישיות, על דעותינו הקדומות ועל 'התחושה החמה' שהדברים אכן הולכים ומשתפרים. בקצרה, עד עכשיו אין דרך מדעית מכובדת שבאמצעותה נוכל להחליט אם השיטות הפורמליות שיפרו את תהליך פיתוח התוכנה.

כיצד נוכל לתקן את המצב? רק על ידי שימוש במודלים של תהליך ובמדידות היבטי המפתח של מודלים אלה, שיאפשרו לנו לערוך ניסויים בגישות שונות ולהעריכן. בעזרת הנחיה מסוימת מצד התיאוריה והמעשה של המדידה, נוכל להתבונן עתה בתהליכים עצמם ולנסות לקבוע כיצד הם פועלים, תוך מגמה לחקור אפשרויות שונות לשיפור התהליכים.

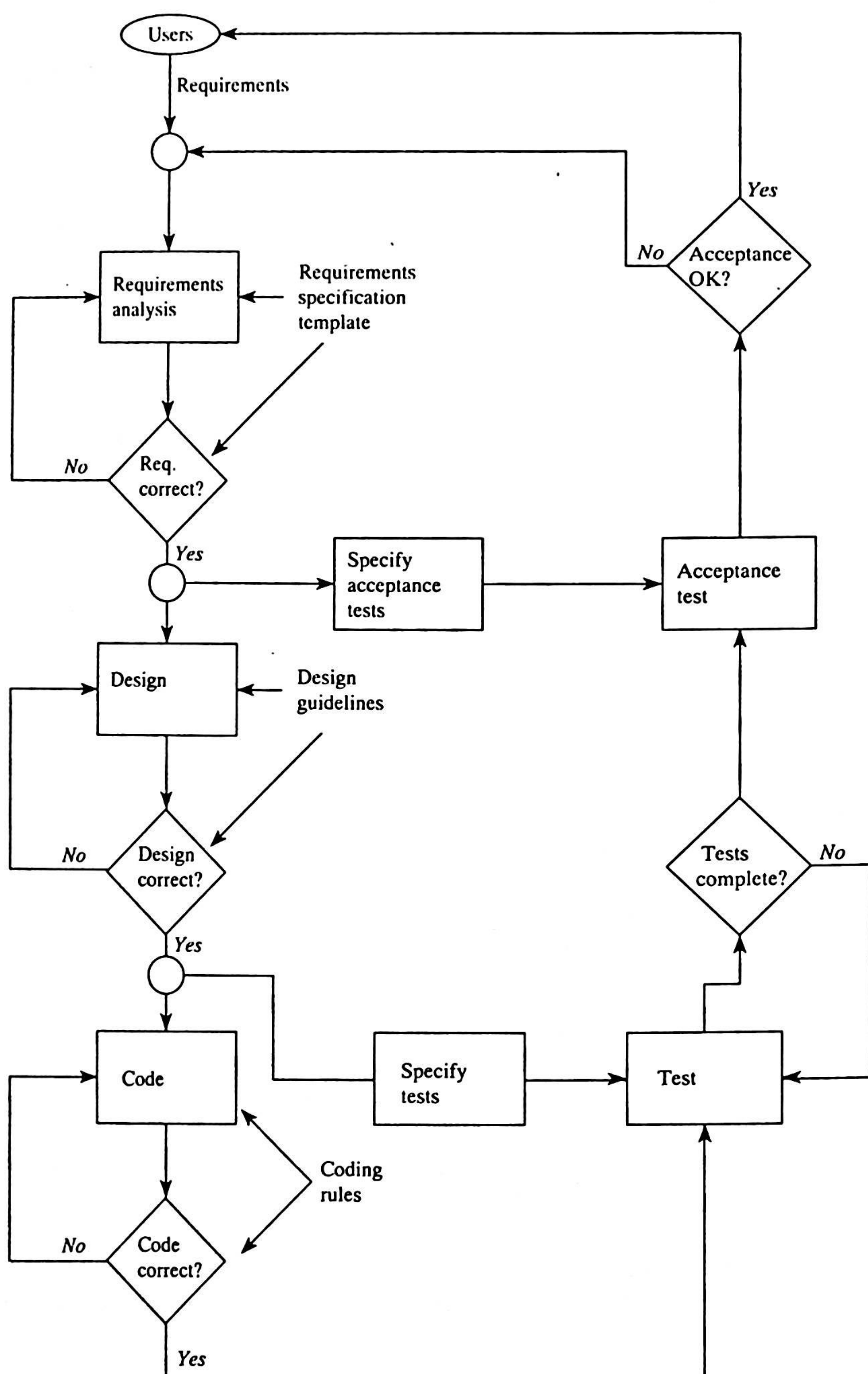
נעשה זאת בשלבים. תחילה נבדוק את הדרכים למידול תהליכים, כדי שנוכל לקבל מושג ברור על הדרך בה פועלים התהליכים. רק אז ננסה לכמת את הביצועים במונחים של משתנים עיקריים שמתגלים על ידי המודל. את ביצועי כל תהליך אפשר למדוד אז כהקדמה לקביעת מטרות לביצוע, ולבחירת אסטרטגיה לשיפור התהליך.

טכניקות למידול תהליכים

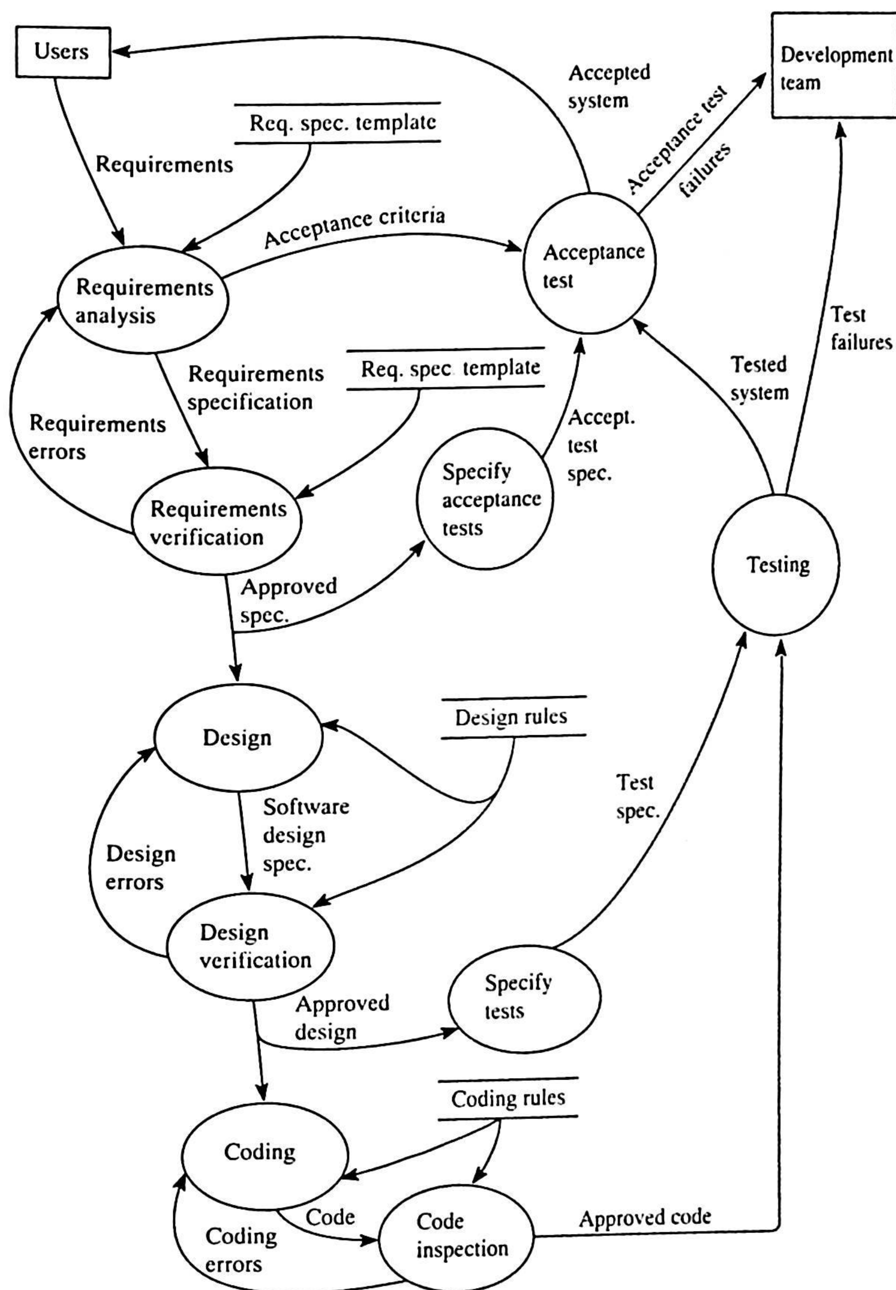
מטרת **מידול תהליכים** (process modelling) היא לעשות את התהליך גלוי לעין ונגיש לבחינה מפורטת. סביר להניח שכלי המידול החזקים ביותר הם אלה שנוסו ונבחנו ביישומים אחרים, כגון תרשימי זרימה (של הפעילויות) ותרשימי זרימת נתונים.

תרשימי זרימה (ראה תרשים 6.6) מתרכזים במבנה הלוגי של התהליך; הם מזהים נקודות החלטה ומתעדים את ההחלטות המעורבות בתהליך. תרשימי זרימת נתונים (ראה תרשים 6.7) מתרכזים בתנועת המידע דרך התהליך וסביב לו, ומאפשרים את ניתוח הפונקציה של עיבוד המידע. בעזרת המראה המבני והפונקציונלי של התהליך, יש ביכולתנו לזהות את התוצאות האפשריות של שינויים במבנהו.

במודל התהליך אפשר להשתמש לזיהוי מבני בקרה, או המרות נתונים הנותנים לתהליך את אופיו. אפשר לראות אם יש דברים המגבילים את ביצועי התהליך, ולזהות את סוג השיפורים האפשריים. 'מנהלי' הביצועים צריכים להימדד כעת, כדי לאפשר תרגול בבדיקות המכוונות לזיהוי הדרך הטובה ביותר לחיפוש ביצועים משופרים.



תרשים 6.6 *תרשים זרימה המייצג תהליך פיתוח תוכנה*



תרשים 6.7 תרשים זרימת נתונים המייצג תהליך פיתוח תוכנה

אסטרטגיות לשיפור תהליכים

מהאמור לעיל משתמע, שיכולות להיות שתי גישות לנושא שיפור התהליכים: מלמעלה למטה (כשמתחילים במודל מוכלל של התהליך), ומלמטה למעלה (כשמתחילים במודל מפורט של הנוהג הקיים בארגון). למעשה, שתי הגישות צריכות להשתלב לאסטרטגיה אחת של שיפור תהליכים.

שיפור תהליכים מתמקד ביעדי הארגון בתהליכים ובמוצרים הכרוכים בהשגת יעדים אלה.

גישת 'מעלה-מטה'

הגישה ***'מעלה-מטה'*** (top-down) מתחילה במודל מוכלל של תהליך פיתוח התוכנה, כגון התהליך המגולם בתקן ISO 9001 ובתקן ISO 9000-3, ומיישמת אותו בארגון כדי לזהות את הפערים הקיימים בין הנוהג הנוכחי לבין דרישות המודל. לאחר זיהוי הפערים, מכינים ומיישמים תוכנית פעולה, שמתאימה את התהליכים שבארגון לאלה של המודל. גישה זו מתרכזת בתהליכי איכות כלליים האמורים להימצא בכל ארגון העוסק בפיתוח תוכנה.

גישת מעלה-מטה מסייעת ביצירת תשתית איכות יציבה, בשילוב עם מדיניות איכות, ונהלים מתועדים שנועדו לכסות תחומים כגון ביקורת פנים, פעולה מתקנת, סקרי הנהלה ופעולות הדרכה. היא לא מספקת סיוע רב ביצירת נהלים לתיעוד התהליכים העסקיים הייחודיים לארגון, כגון תהליכי פיתוח תוכנה, או תהליכי תמיכה ללקוחות.

בעוד שהמדידה היא חלק בלתי נפרד מיישום התקנים ISO 9001 ו- ISO 9000-3, מסמכים אלה אינם מגדירים במפורש את אסטרטגיית שיפור התהליכים. יישום התקנים נוטה אם כן, לעודד ארגונים לשפר את תרבות האיכות שלהם רק עד לרמה הנדרשת להשגת ***ההסמכה*** (certification), אך אינו תורם לשאיפה שמעבר לרמה זו.

גישת 'מטה-מעלה'

הגישה 'מטה-מעלה' (bottom-up) ממוקדת יותר במצב המקומי. היא מתחשבת במאפיינים הייחודיים והמיוחדים של הארגון, בתהליכים ובמוצרים שלו. לגבש אסטרטגיית שיפורים שתפעל עבור הארגון המסוים ותגיע לאופטימיזציה של תרבות הארגון וקבוצת התהליכים המיוחדת לו.

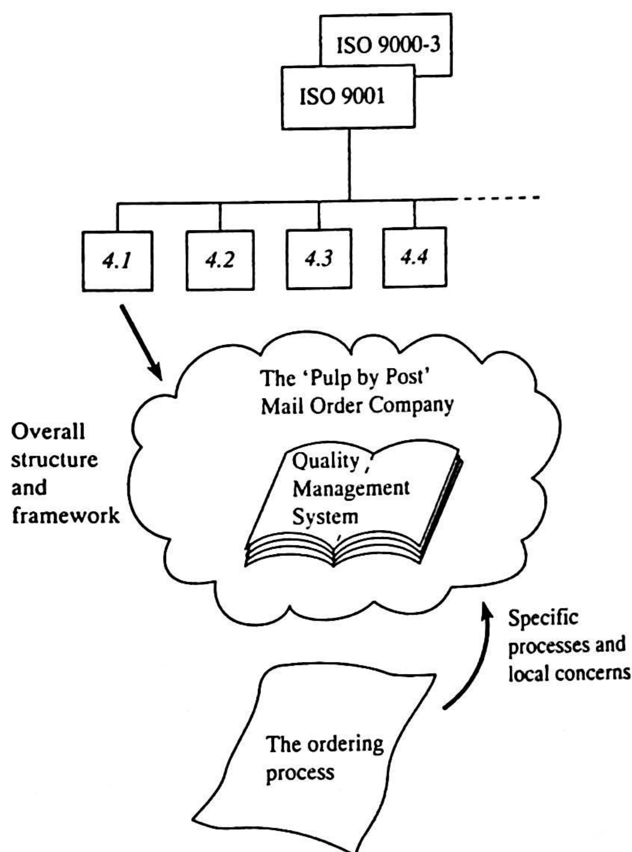
גישת מטה-מעלה מתאימה הרבה יותר לניתוח ולתיאור תהליכים אפקטיביים שיבצעו את פעילויות המפתח בארגון. גישה זו אינה מוליכה בהכרח למערכת איכות מאוזנת, בה ייכללו מנגנוני הניטור והמשוב הדרושים כדי לוודא שהמערכת תישאר אפקטיבית לטווח הארוך.

אופייה היפתוחי של גישת מטה-מעלה לא מציע גבולות כלשהם בנושא שיפור התהליכים. באותה מידה, הוא גם אינו מספק כל אמצעי ביטחון כנגד הסחף בתרבות האיכות.

אסטרטגיה משולבת

יתרון אחד הקשור בשימוש בתקנים ISO 9001 ו- ISO 9000-3 הוא העובדה שהשימוש בהם אינו בעל אופי מחייב. יש מרחב פעולה גדול לאופטימיזציה מקומית, אך באותה עת, יש בו גם דרישה עקבית להפעלת תהליכי מפתח מסוימים. אסטרטגיית מעלה-מטה המבוססת על תקן ISO 9000-3 אינה שוללת את השימוש בקבוצת תהליכים המותאמים ביותר לארגון; אדרבא, היא אפילו מעודדת גישה זו.

עם זאת, פעולה על פי תקנים ISO 9001 ו- ISO 9000-3, יכולה להצעיד את הארגון רק עד למרחק מסוים. עבור השגת שיפורים נוספים, יש צורך בתוכנית שיפורים המוגדרת ומותאמת בצורה האופטימלית ביותר לצרכים המקומיים. היא משתמשת במבנה הבסיסי של ISO 9001 ושל ISO 9000-3 כבנקודת מוצא, ומוסיפה אסטרטגיה לשיפור תהליכים המבוססת על מדידות, כמתואר בתרשים 6.8.



תרשים 6.8 מנגנונים לשיפור תהליכים

מדדים מעשיים לשיפור תהליכים

אין ספק שהדרך הטובה ביותר בה אפשר להחליט מה הן המדידות הדרושות לניהול תרגיל לשיפור תהליכים, היא לבחון את מודל התהליך ולקבוע יעדים לשיפור. אחר כך, ניתן לחוקר אותם בגישת GQM (שאלת מדד היעד). מדידות המתגלות בניתוח זה תמקדנה את תשומת הלב בדיוק על המשתנים העיקריים בתהליך הנמצא בבדיקה.

לדוגמה, כאשר משתמשים במחזור החיים מסוג V, אנחנו יכולים לחשוש שתרגיל ניתוח הדרישות לא יהיה מספק, משום שבזמן מבדק הקבלה אנו מוצאים הרבה יותר שגיאות מהצפוי, אף על פי שמבדקים אחרים מסתיימים בהצלחה. כיצד נוכל לתקוף את הבעיה האמיתית? היעד שלנו הוא צמצום מספר השגיאות המתגלות בזמן מבדק הקבלה. השאלות העיקריות שיעסיקו אותנו יהיו בוודאי 'האם אין ביכולת ניתוח הדרישות לזהות את כל הדרישות?'; 'האם מבחני טרום הקבלה אינם מצליחים לגלות שגיאות?'; 'האם מבחני הקבלה הם באמת בבואה נאמנה של הדרישות?' שאלות אלו ניתנות לפירוק לשאלות פשוטות יותר, כגון: 'מהו סוג השגיאות שאנו מגלים?'; 'מהו התחום הפונקציונלי של המערכת שמפיק את רוב השגיאות?'; 'לאילו מהתוכניות ניתן לייחס את השגיאות שהתגלו?' וכן הלאה. לאחר שנדע את התשובות לאחדות מהשאלות הדומות לאלו שפורטו לעיל, נהיה בעמדה שתאפשר לנו להטיב לזהות ולסלק את הסיבות השורשיות.

כאשר התהליך עוסק בפיתוח תוכנה, אנו יכולים להשתמש בניסיון העבר שיספק לנו עמדת זינוק לניתוח התהליך. פעילות מידול התהליך היא בכל זאת חיונית לאפקטיביות לטווח ארוך של שיפור התהליכים. עם זאת, אפשר יהיה ללמוד אודות התהליך שלפנינו במידה שתאפשר לנו לפתוח בהתחלה נסיונית בכיוון אסטרטגיית השיפורים, עוד לפני סיום פעולת המידול, על ידי בחינה של מספר מדידות שנתגלו במספר משמעותי של סביבות שונות ובמודלים שונים של מחזור החיים. דבר זה יכול לספק מידע מסוים שניתן יהיה לבסס עליו את ניתוח GQM ההתחלתי. לפנינו מבחר מדידות שמן הראוי לנסותן בתחילת הפעילות.

מבחר של מדידות שימושיות

1. **פרופיל תפעולי**

הפרופיל התפעולי הוא מדידת השכיחות של כל פונקציה במערכת, שנמדד בדרך כלל על ידי ניטור הבחירות מתוך התפריט. פרופילים תפעוליים חזויים יכולים לתאר באופן שימושי את הדרך בה צפוי שנשתמש במערכת. פרופילים תפעוליים מציאותיים יכולים לסייע בקביעת סדר עדיפויות לעבודות התחזוקה והבדיקות.

2. **חומרת האירועים**

חומרת האירועים היא מתכונת לסיווג האירועים המדווחים, בה אירוע הוא כל דבר הגורם בעיות למשתמש (לא בהכרח שגיאת תוכנה). מדידת האירועים לפי מידת החומרה תסייע למקד את פעילויות השיפורים. הדרכה, או שינוי נהלים יכולים להועיל יותר מאשר תיקון שגיאות סתם.

3. **ממדי הבדיקות**

היקף הבדיקות הוא ערך המראה איזה חלק של התוכנית נבדק כבר. מדידת היקף הבדיקה באמצעות משפטי תכנות ופקודות הסתעפות מגלות עד כמה היו הבדיקות אפקטיביות בהפעלה של הקוד. נושא זה נדון בפרק 4.

4. **שגיאות שדווחו**

ספירה פשוטה של שגיאות שדווחו במערכת מסוימת נותנת מדידת קו-בסיס של מצבה הנוכחי של התוכנה. השוואת השגיאות המדווחות על בסיס שבועי, או חודשי מספקת מדידה של היציבות, ומסייעת לגלות מגמות כלשהן.

5. **שגיאות בזמן הסקר החוזר**

שגיאות שנתגלו בזמן הסקר מספקות אומדן גס לאיכות המוצר המוגמר. שגיאות במסמך התיכון מספקות הכוונה לבעיות שיש לצפות בשעת הבדיקות, ספירה גבוהה של שגיאות שנתגלו בזמן הסקר מראה לעיתים תכופות על סיכויים רבים לאחוז דומה של שגיאות בזמן הבדיקה. ניתוח שגיאות שמתגלות בזמן הסקר יכול לשמש לזיהוי שיפורים בתהליך הפיתוח. נקודה זו משמשת נושא לדיון רחב יותר בהמשך.

6. **שגיאות בזמן הבדיקה**

שגיאות בעת בדיקות לפי שלבים, מציגות מדד של מצב המוצר תוך כדי התקדמות הבדיקות. בשלבים המוקדמים, ההשוואות השבועיות מגלות את מידת יציבות הקוד, דבר שהוא גורם חיוני בהחלטה מתי ניתן להתחיל בשילוב החלקים השונים של התוכנה.

7. **תיקוני שגיאות שהושלמו**

ספירה שבועית של תיקוני שגיאות שהושלמו, מוסיפה למדידה של שגיאות בזמן הבדיקה. כאשר מציגים אותן באותה מערכת צירים, מספקות השגיאות שדווחו והשגיאות שתיקוניהן הושלמו, תמונה טובה וכוללת של יציבות התוכנה ומסייעות להחליט אם יש צורך במשאבים נוספים בצוות הפיתוח.

8. **המאמץ ומשך המשימות**

מדידות המאמץ ומשך המשימות השונות, דרושות עבור בקרת הפרויקט. עם זאת, הן מספקות גם ערכים באמצעותם ניתן לכייל את מודל מחזור החיים המשמש לתכנון הפרויקט. התרעות מוקדמות אפשריות גם כן על ידי ניטור מגמות המעידות על כך שמוקדשת תוספת זמן ומאמץ לפעילויות התיכון.

9. **עלות על פי סוג שגיאה**

מדידות העלות לתיקון כל סוג שגיאה מדווחת, מספקות משוב לתהליך הפיתוח ומצביעות על המקומות בהם יש צורך רב יותר בשיפורים. מדידות אלו יכולות לסייע בקביעת סדר העדיפויות בעבודת תיקון השגיאות.

10. **שגיאות לתוכנית**

ספירת שגיאות לפי תוכנית יכולה להיות שימושית בחשיפת תוכניות שיש מקום להחליפן או לעצבן מחדש.

מבדקים ומדידת ביצועים

מבדקים (benchmarking) או **השוואת ביצועים** ראויים לאזכור מיוחד, משום שזו טכניקה המכוונת במיוחד לקביעת יעדי שיפורים והשגתם. בספר זה איננו מעוניינים רק במבדקי תחרות, כלומר, קביעת יעדים המבוססים על ביצועי הטובים שבמתחרינו. אנו מעוניינים גם בעיקרון הבסיסי של קביעת יעדי ביצועים, מבוססים על מידע אובייקטיבי של מה שהוא בר-הישג, או הושג בפועל, ולא להשתמש ביעדים שרירותיים של אחוזי השיפור.

למשל, נוכל להציב לנו יעד: להקטין בשנה הבאה ב-10 אחוזים את כמות התקלות המדווחות על ידי הלקוחות. אך אם כל מתחרינו עוברים אותנו בלמעלה מ-10 אחוזים בתחום זה וביצועיהם משתפרים ב-15 אחוז לשנה, הרי שאנו צפויים לצרות. אם ברצוננו להיות 'הטובים בכיתה', עלינו להגביה את רף היעד כך שנוכל לא רק להתאזן עם המתחרים, אלא אף לעבור אותם. זהו הבסיס למבדקים תחרותיים. עם זאת, אנו עלולים להחטיא את היעד שקבענו, אם המתחרים כבר השיגו אותנו. הניסיון לעבור אותם רק יביא אותנו לרפיון ידיים.

אנו זקוקים לגישה שתספק לנו סדרת מבדקים, כדי שנוכל לבחור באסטרטגיה שתוביל אותנו לרמת ביצועים הישגיים באמצעות סדרת צעדים בני-הישג בפרק זמן מציאותי.

קיימות ארבע טכניקות הנמצאות בשימוש נרחב:

1. **מבדקים פנימיים**, שעורכים השוואות בין חלקי אותו ארגון ומשתמשים במוצלחים ביותר כבקנה מידה שהאחרים אמורים להשתדל להשיג אותו.
2. **מבדקים תחרותיים**, שמשווים ביצועים עם ארגונים תחרותיים, במיוחד עם המוליכים בשוק, או עם אלה שיש להם יתרון תחרותי מסוים.
3. **מבדקים פונקציונליים**, שמשווים פעולות פונקציונליות מסוימות עם ארגונים אחרים העוסקים בתחומים פונקציונליים דומים החברות המשמשות להשוואה אינן חייבות להיות בהכרח חברות מתחרות. הן יכולות להיות חברות הידועות כאפקטיביות ביותר באותו תחום פונקציונלי בענף תעשייתי אחר.
4. **מבדקים כלליים**, שמשווים תהליכים עסקיים שלמים עם חברות אחרות (לאו דווקא מאותו ענף), שמשתמשות באותו תהליך בסיסי. בדרך כלל התהליכים חוצים גבולות פונקציונליים אינדיבידואליים.

לכל אחת משיטות אלו יתרונות ומגבלות. בין התחומים הקשים ביותר ניתן לציין את איסוף המידע השימושי אודות המתחרים, ואת הקושי הכרוך ביישום רעיונות חדשים, או מעניינים הלקוחים מענפים אחרים. בתור מפתחי תוכנה, נוכל להשתמש בטכניקות אלו בדרכים שונות:

- על ידי השוואת פיתוח תוכנה עם פונקציות פנימיות אחרות בארגון שלנו;
- על ידי השוואת כלל הפעילות העסקית שלנו עם גוף מתחרה, כאשר שנינו נמצאים בעיסוק הראשוני של פיתוח תוכנה. כלומר, בית תוכנה אחד המשתמש בבית תוכנה אחר לקיום מבדקים תחרותיים;

- על ידי השוואת מחלקת IT (טכנולוגית המידע) שלנו עם זו של חברה אחרת, ואולי בענף תעשייתי אחר;
- על ידי השוואת תהליך פיתוח התוכנה שלנו מול התהליך הקיים בחברה אחרת שעוסקת בפיתוח תוכנה בתנאים דומים.

האפשרות שנבחר תקבע במידה רבה על ידי סוג העסק שלנו, אם כי יכולות להיות לפעמים כמה אפשרויות בחירה.

מודל המערכת לניהול האיכות שמסופק על ידי התקנים ISO 9001 ו/או ISO 9000-3 הוא, למעשה, צורה מסוימת של מבדקים. היא מעידה שכל הארגונים שנרשמו בהצלחה (שעמדו בתנאי התקנים) והגיעו לסטנדרט משותף של ניהול האיכות. עם זאת, תהליך הרישום אינו מודד את הביצועים במובן הרחב יותר, ולכן אינו שימושי כגורם השוואתי אלא רק בזירת האיכות.

עם זאת, יש סכימות כדוגמת **'פרס האיכות האירופי'** (European Quality Award) שמיישמות מודל סטנדרטי של הארגון ומודדות לפי פרמטרים שונים, כולל ביצועים. ארגונים המשתמשים באותו מודל ניתנים להשוואה ישירה. אחת השיטות לעריכת מבדקים יכולה להתבסס על הצטרפות לארגון כגון "המוסד האירופי לניהול האיכות" (EFQM) ולהשתתפות במידע עם ארגונים אחרים, שאינם בהכרח מתחרים ישירים, בכל הנוגע לביצועים בהשוואה למודל של EFQM (הערה: ספר זה נכתב באירופה ומתייחס למצב הקיים שם, גם בישראל יש מודל ל**פרס האיכות בתעשיה**, המנוהל על ידי איגוד תעשיות האלקטרוניקה, ראה נספח).

מבדקי פיתוח תוכנה

פיתוח התוכנה הוא פעילות המשותפת לארגונים רבים מתחומים עסקיים שונים. מכיון שכך, יכולים להימצא ארגונים רבים שעוסקים בפעילויות פיתוח הניתנות להשוואה, ושאינם בהכרח מתחרים. עריכת מבדקים יכולה להיות פעילות פורייה, ועם זאת אין בה סיכונים הכרוכים בהתקרבות יתר אל המתחרה.

דרך טובה להשגת רמה שימושית של שיתוף במידע היא על ידי יצירת **'מועדוני מבדקים'** ('benchmarking clubs'). החברים במועדונים אלה מחליפים ביניהם מידע על התהליכים והביצועים שלהם לתועלת כל הנוגעים בדבר. מועדוני מבדקים קיימים כבר במגזרים תעשייתיים מסוימים. מועדון מבדקים לתוכנה יכול להיות פתוח לכל ענפי התעשייה, והנושא המשותף יכול להיות מידת המעורבות של ההנהלה בפיתוח תוכנה. על החברות במועדונים כאלה ותועלתה אפשר ללמוד מ'מועדונים' קיימים, כגון האגודה הבריטית - British Computer Society Specialist Interest Group, או איגודים אחרים במישור הבינלאומי.

מה ישמש נושא למבדקים? אנו יכולים להעלות על הדעת מספר מדידות. **ביצועים** יכולים להימדד במונחים של מספר שגיאות מול מספר שורות קוד, או על ידי היחס שבין מספר השגיאות לבין מספר הפונקציות שהושלמו, כפי שנמדדו על ידי הלקוחות. ניתן להעריך את **הפריון** במונחים של מאמץ הפיתוח מול שורות קוד, או מול פונקציות

שסופקו. **איכות** התוכנה שנמסרה יכולה להימדד גם על ידי מדידת מאמץ התחזוקה המוקדש אך ורק לפעילויות תיקון שגיאות. את **האפקטיביות** אפשר להעריך על ידי מדידת היחס של מצביעי הצלחה קריטיים שהושגו שישה חודשים לאחר האספקה. זהו מבדק שימושי במיוחד במקרה שנקטנו בצעדים לשיפור תהליכים, כדי לצמצם את מספר השגיאות שיתגלו במבחני הקבלה.

נראה שהמדידות עליהן הצבענו גסות למדי, ובפועל, מדידתן תהיה שונה מארגון לארגון. לכן, ערך ההשוואות הפשוטות יהיה מוטל בספק רב בתחילת כל תרגיל מסוג זה. עם זאת, יצירת פרמטרים סטנדרטיים שיימדדו באותן הדרכים בארגונים השונים תוכל להביא תועלת רבה במרוצת הזמן. השקעת זמן וכספים יוצרת בסיס לפעולת המבדקים, ויחד עם זאת, מעשירה את נהגי המדידה בכל הארגונים המשתתפים בפעולה זו.

מדידת ביצועי פיתוח התוכנה

הדבר הראשון בו יש להכיר בתחום מדידת הביצועים של כל תהליך, הוא שהמדידה מתחילה להיות בעלת משמעות רק כאשר התהליך הנמדד הוא יציב. כלומר, שהתהליך בשימוש רציף וסדיר על ידי כלל העובדים השייכים לנושא, ושהתהליך הבסיסי אינו נתון לשינויים יסודיים, או תכופים. מכל זה ברור, שהתהליך **מתועד** גם בכתב.

בעיקרון, מנגנון שיפור הביצועים פשוט למדי.

מרשם פשוט לשיפור ביצועים

1. החלט על הפרמטרים העיקריים של התהליך.
2. מדוד את כל הפרמטרים האלה ורשום את ערכיהם.
3. החלט על יעדים עבור כל פרמטר.
4. בחר באסטרטגיות שיפורים מתאימות לשיפור של אחד, או יותר מהפרמטרים האלה.
5. הפעל טכניקת שיפורים אחת שמכוונת לשיפור פרמטר אחד.
6. עקוב אחר הביצוע עד שתגיע למצב יציב חדש.
7. אמוד את הערך של השיפור המוצע.
8. אם הצלחת, השאר את השיפור במקומו והפעל שיפור חדש לגבי פרמטר אחר.
9. בכל שלב, אם אחת מדרכי השיפור אינה מצליחה לשפר את הפרמטר שבחרת בו, או שיש לכך השפעה לא רצויה על פרמטר אחר, עצור ונתח את המצב לפני שתמשיך.
10. המשך עד שכל אפשרויות השיפור שבחרת תופעלנה במשולב.
11. קבע לעצמך יעדים חדשים.

ההחלטה החשובה ביותר שעליך לקבל מתייחסת לבחירת הפרמטרים העיקריים. טבעי שהפרמטרים העיקריים עבורך מותנים במחזור החיים המסוים של הפיתוח שלך ובתרבות העסקית. עם זאת, יש כמה קווי פעילות משותפים שתרמו להבנה רבת ערך אצל מפתחים רבים.

רוב האנשים מתחילים במדידת פגמים. אלה קלים יחסית למדידה, ובדרך כלל מספקים תמונה די ברורה של מידת ההצלחה בהימנעות משגיאות, או סילוקן מתהליך הפיתוח. זהו בדיוק מה שתקן ISO 9000-3 מצפה ממך, ולכן כדאי הדבר, ולו רק מהבחינה הזו. עם זאת, מדידת הפגמים מגלה הרבה בכל הנוגע ליציבות תהליך מחזור החיים שאתה נוהג לפיו, ובדבר 'תרבות האיכות' שלך. יהיה עלינו להרחיב יותר את הדיבור בשתי סוגיות אלו לפני שנחזור לנושא מדידת הפגמים.

היציבות נחשפת מייד על ידי נסיונותיך להקים את תהליך המדידה עצמו. לעיתים תכופות מתעוררות בעיות ברגע שנודע לאנשים שיש תוכנית למדוד תפוקת שלב מסוים של מחזור החיים. מתעוררים ויכוחים על המקום המתאים ביותר למדידת הפרמטר הנדון, או על המשמעות הנכונה של הפרמטר, אם יש לו בכלל משמעות. ויכוחים מסוג זה יעכבו את תוכנית המדידה שלך, אך אל תתפתה לקיצורי דרך כדי לפצות על הזמן שאבד. הוויכוחים הם חיוניים. עדיף היה אילו הם היו מתעוררים קודם לכן, אך עתה אסור להמשיך בלעדיהם. משעה שמסתיימים הוויכוחים, יש לפניך תהליך יציב ובסיס טוב יותר לשיפור תהליכים, שלא לדבר על הערך הרב של התרגיל בתקשורת, שתרם לפתרון הבעיות.

תרבות האיכות היא אותה נטייה אמיתית, אך בלתי מוחשית, לשיפור תהליכים. ארגון בעל מחויבות ושואף קידמה בדרך כלל תופס בשתי ידיים כל הזדמנות לשיפור תהליכים ומצעיד אותה קדימה בעוז. ארגון בעל מחויבות נמוכה יותר ופחות נמרץ, מוצא מכשולים ותירוצים לחוסר הקידמה, בהתאם לרמת המחויבות שלו. ייתכן שספירת הסיבות הניתנות לאי נקיטת פעולות לשיפור תהליכים היתה יכולה לשמש כלי מדידה טוב למידת המחויבות לאיכות.

מדוע לעסוק בפגמים? הפגמים מאפשרים לנו מבט לאחור אל תוך תהליך הפיתוח, ויכולים לחשוף חולשות. נוסף לכך, יש בהם כדי להשליך אל התקופה של השימוש התפעולי ולנבא את הבעיות ואת רמת התמיכה שעשויה להידרש. ספירת הפגמים יכולה להיות פשוטה בתנאי שתנהל רישום של האירועים ותנסה לסלקם (אם לא תעשה זאת, מעטים הסיכויים שיהיה לך תהליך המסוגל להשתפרות שיטתית). ניתוח הנתונים שאתה אוסף הוא קצת יותר מסובך, ובקטע הבא נעסוק בו ביתר פירוט.

ספירת הפגמים אינה הפרמטר היחיד שיש בו עניין, ולא תמיד הפרמטר החשוב ביותר בכל המקרים, עם זאת, הוא משמש כאן כדוגמה פשוטה לדרך בה אפשר להתחיל בתרגיל שיפור התהליכים. ככל שתמשיך, תמצא בוודאי שמתגלים פרמטרים אחרים המתחרים על תואר 'הפרמטר החשוב ביותר'. שיפור תהליכים הוא תהליך איטרטיבי, והאסטרטגיה האפקטיבית ביותר לטווח ארוך היא לנסות לבצע שינויים קטנים על בסיס מתמשך. מדידה וניתוח נתוני הפגמים הם דוגמה טובה לאחד השיפורים הקטנים האלה.

שיפור תהליכים מעשי

הצבת מטרות מציאותיות לשיפורים

כשאתה מודע לבעיות כבדות, סביר שתרגיש שכל שיפור מביא לך תועלת. אם לעומת זאת, יש לרשותך תהליך יציב ומפותח היטב, שמזה זמן אתה מקפיד לשפרו, יש להניח שתהיה מעוניין בלא פחות מאשר עריכת מבדק (benchmark) שישווה בינך לבין מפתח התוכנה היעיל ביותר שתוכל למצוא. רוב הארגונים נמצאים בין שני קצוות אלה.

מה אם כן יכול להוות יעד סביר?

שבעה צעדים קלים כדי להיות 'הטוב ביותר'

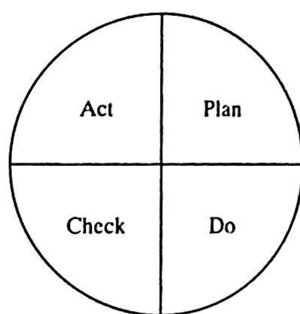
1. הקם תהליך יציב ומתועד. החלט על הפרמטרים העיקריים.
2. ערוך ביקורת של התהליך מול הנהלים המתועדים, ותקן את כל הפגמים שייחשפו.
3. מדוד את הפרמטרים העיקריים, קבע קו-בסיס לביצועים, ופעל לגבי ליקויים בביצוע.
4. קבע מטרות שרירותיות לשיפורים (נניח 5 עד 10 אחוזים), כדי להחליט עד כמה קל או קשה לבצע שיפורים.
5. קבע יעדים קשים אך ניתנים להשגה, ושפר בהתמדה את התהליך עד להשגתם.
6. קבע יעדי מבדק חיצוניים ושפר בהתמדה את התהליך עד להשגתם.
7. קבע יעדים למבדק של 'הטוב ביותר' ושפר את התהליך, עד שתעבור את היעדים שקבעת, באחוז שוליים גדול מהשיפורים שאתה מצפה שיבצעו מתחריך.

תוכל להתחיל את תוכניתך לשיפור התהליכים מכל נקודה שנראית לך מתאימה ולסיימה בכל נקודה שהיא (או כאשר ייגמר לך המרץ).

מחזור PDCA

מחזור PDCA מוצג בתרשים 6.9. הוא מייצג את המראה הפשוט ביותר האפשרי של תהליך שיפור התהליכים: Act ,Check ,Do ,Plan (תכנן, בצע, בדוק, עשה). תכנן את השינויים שאתה מצפה לבצע ואת דרך הביצוע. בצע את השינויים. בדוק את אפקטיביות השינויים על ידי מדידת הפרמטרים המתאימים. פעל על פי המדידות, בכך שתחזור על התהליך.

התבוננו כבר בחלקי הביצוע והבדיקה. עתה דרושים לנו אמצעים אפקטיביים כדי להחליט איזה שינויים לבצע.



תרשים 6.9 *מחזור PDCA*

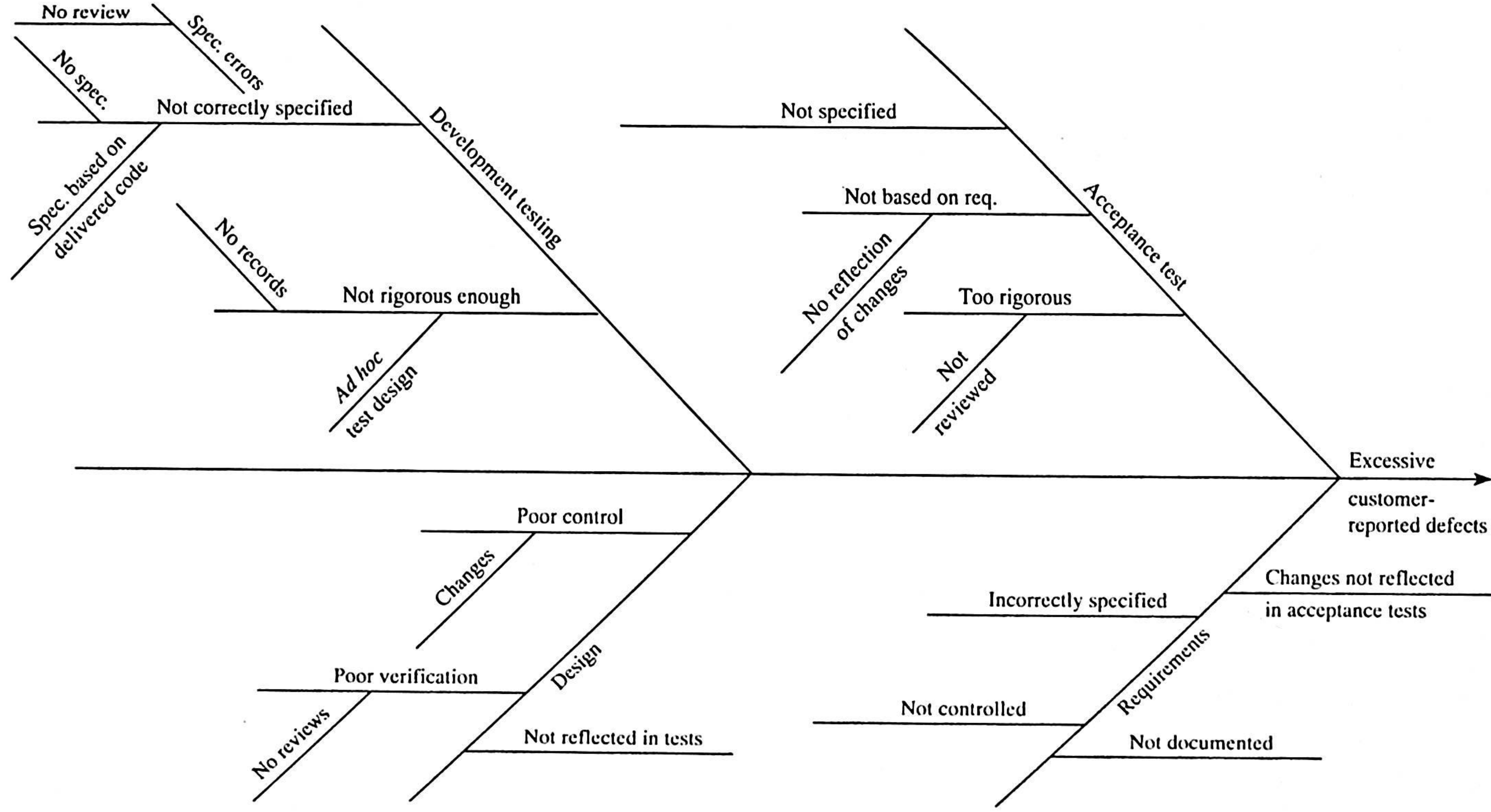
טכניקות וכלים

נחזור אל תרחיש ספירת הפגמים. נניח שאנו מספקים תוכנה שיש בה פגמים בשיעור של 35 פגמים בשנה, לפי דיווחי הלקוחות. האם זה מספר משמעותי? מה מידת חומרת הפגמים? איזה מוצרים מייצרים את המספר הגבוה ביותר של פגמים? כמה עולה לתקן את הפגמים? מדידה אחת מובילה לשאלות חדשות שמחייבות מדידות נוספות.

כאשר אנו מציבים לעצמנו את המטרה הפשוטה של צמצום מספר הפגמים ב-10 אחוז, נוכל להשתמש בגישת GQM (יעד, שאלה, מדד) ולעורר מספר גדול ככל האפשר של שאלות. במקום זאת, נשתמש בפגמים אלה כדי לחקור את התהליך עצמו ביתר פירוט. כדי לפשט את חקירותינו, נבחר במקרה בו אנו מספקים ותומכים במערכת אחת בלבד, הנמצאת בשימושו של ארגון אחד, ובו רק משתמש אחד.

מניין צצות כל השגיאות האלו? אחת הדרכים בהן נוכל לברר זאת, היא 'מכשורי' התהליך על ידי מדידת פגמים בכל שלב. שגיאות שהתגלו בסקר התיכון, שגיאות שהתגלו בבדיקת יחידות, שגיאות שהתגלו בזמן בדיקות שילוב היחידות, ושגיאות שהתגלו בזמן בדיקת המערכת, כל אלו מספרות לנו דבר מה על התהליך. היכן מתגלות רוב השגיאות? מהו סוג השגיאות שמתגלה בכל שלב? כדי לנהל חקירה ממין זה, אנו זקוקים להגדרות ברורות של מה שאנו מחפשים בכל שלב, של מה שאנחנו מגדירים כפגם. אנו גם זקוקים למנגנון לסיווג הפגמים שנגלה, כדי שנוכל לקבוע מהי מידת החומרה של כל פגם.

דרך אחרת בה נוכל לחקור את מקורות הפגמים היא לנסות, בשיטת סיעור מוחות, להעלות על הדעת סיבות אפשריות לפגמים. לאחר מכן, צריך לחקור את הפגמים המדווחים, ולשייך לכל אחד מהם את אחת הסיבות שהעלינו בדעתנו. פגמים שעבורם לא יימצאו סיבות מתאימות יפתחו פתח למציאת סיבות לקטגוריות נוספות של סיבות. כך לא יקשה עלינו לזהות מהן הסיבות הפוטנציאליות הגורמות לרוב התקלות, ולטפל בסיבות אלו בעדיפות גבוהה ביותר להשגת שיפורים.



תרשים 6.10 *דיאגרמת אישיקאווה לבעיית שגיאות יתר במבחני הקבלה*

תרשים 6.10 הוא דוגמה של **דיאגרמת אישיקאווה** (Ishikawa Diagram) לבעיה, כאשר הלקוחות מדווחים על שגיאות רבות מאוד. דיאגרמה זו מהווה מנגנון יעיל ביותר להקמת מבנה לפעילות של סיעור מוחות ולרישום התוצאות. הדיאגרמה נבנית על ידי העברת קו אופקי יחיד המייצג את הבעיה, כגון רמה גבוהה של פגמים. המשתתפים משמיעים את כל הסיבות האפשריות. הסיבות הנראות כסבירות ביותר נדונות ומוערכות, והמשמעותיות ביותר נרשמות בדיאגרמה בצורת קווים המסתעפים מהקו המרכזי בדומה לעמוד שדרה של דג. לפיכך, דיאגרמות אלו נקראות לעיתים בשם '**דיאגרמת אידרת הדג**' (fishbone diagram). לאחר זיהוי הרמה הראשונה של הסיבות האפשריות, אפשר לנתח כל אחת מהן ניתוח נוסף באותו תהליך, כדי לזהות את הרמה השנייה של סיבות אפשריות. בדרך זו, ניתן לבנות עץ שלם של סיבה-תוצאה ולנתח אותו, כדי לקבוע את הסיבות השורשיות לבעיות.

זהו ניתוח חשוב מאוד. מאמץ שיושקע בסילוק סימפטומים, או אפילו סיבות מהרמה הראשונה, לא יסלק, בדרך כלל, את הבעיות. בדרך כלל, חוזרים על כל התהליך עד שלבסוף מתגלות הסיבות השורשיות. אם מצליחים להכיר את הסיבות השורשיות ולסלק אותן כבר בשלב אחד, חוסכים מאמץ רב בהוצאות ובזמן.

דוגמה נוספת לניתוח מסוג שגיאה וסיבה ניתנת בתרשים 6.11. זהו יומן שגיאות שהתקבל מסקר התיכון. יומן השגיאות נוצר על ידי ספירת השגיאות שדווחו בסוף סקר התיכון והקצאתן לאחת, או יותר מהסיבות האפשריות. השגיאות סווגו לסיבות ראשיות ומשניות; שגיאה ראשית הוגדרה כשגיאה שיכולה לגרום לכשל של תוכנית, ואילו שגיאה משנית הוגדרה כשגיאה שלא תגרום בהכרח לתפקוד לקוי של התוכנית, אך היא פוגעת באחד הכללים. חריגה מדיסציפלינת התכנות למשל, תסווג כמשנית.

סקר מפרטי התיכון	**ראשי**			**משני**		
	חסר	**שגוי**	**נוסף**	**חסר**	**שגוי**	**נוסף**
DFD לא עקביים	2	1				
שגיאה במבנה הנתונים		2				
תיכון שאינו עקבי למפרטים	1		2			
חסר מפרט המודול	1					
חסרה זרימת הנתונים	2					
מפרט מודול שאינו תואם לתהליך ב-DFD		4				
הגדרת מבנה הנתונים לא תואמת את DFD		2	1			
DFD לא מדורג נכון					2	
מילון נתונים לא מעודכן	1					
הערה לא מספיקה להבהרת מפרטי המודול				4		
סיכומים	7	9	3	4	2	-

ראשי: *שגיאה שמובילה לכשל במערכת;* ***משני:*** *שגיאה בתיעוד, או בהיצמדות לכללים;*
Data Flow Diagram - DFD - תרשים זרימת נתונים

תרשים 6.11 *יומן שגיאות תיכון*

השימוש במונח 'משני' אולי אינו התווית ההולמת ביותר שגיאות אלו. נכון שמתייחסים אליהן לעיתים כאל שגיאות פשוטות ומובנות, אך לפעמים, תוצאותיהן יכולות להיות הרות אסון. כל שגיאה 'משנית' מייצגת בעיה פוטנציאלית עבור צוות התחזוקה שיצטרך לתמוך במוצר, תוך כדי מתן השרות. שגיאה 'משנית' כדוגמת פגיעה בדיסציפלינת התכנות עלולה להוליך מאוחר יותר לבעיות ביישום שינויים שהם פשוטים למראית עין. באותה דרך, מחדלים 'משניים' בתיעוד יכולים להותיר את עובדי התמיכה במצב בו יהיה להם מידע מועט מדי למעקב אחר בעיה. כל שגיאה 'משנית' יכולה להפוך לבעיה ראשית מרגע שמכניסים את התוכנה לשימוש. אין לשכוח שתקופת השירות הניתנת לתוכנה, היא בדרך כלל, ממושכת יותר מאשר תקופת הפיתוח שלה.

שגיאות מסוימות שכיחות יותר מאחרות, ויש שגיאות שקל יותר לזהות מאשר אחרות אך אף שגיאה אינה פשוטה (משנית) באמת.

סיווג השגיאות יהיה מבוסס על ניסיון אישי, ושונה לגבי כל סוג מסמך שתסקור. מה שיילכד בסיווג השגיאות הוא מידע אודות סוגי השגיאות החשובים ביותר, אשר תרצה לעקור מן השורש. בסקר התיכון שבדרג העליון, בדרך כלל, ישקפו השגיאות סוגיות הנוגעות לשלמות ולעקביות, בעוד שברמת התוכנית הן תתייחסנה (בדרך כלל) לשימוש הנכון והלא נכון במבני תכנות מסוימים, או לטיפול נכון בסוגי נתונים מסוימים. המידע שביומן השגיאות יכול לשמש אותך לפחות בשלוש דרכים: שיפור התהליך כלפי מעלה; שיפור התהליך כלפי מטה; ושיפור תהליך סקר התיכון.

ראשית, **תהליך הסקירה** (review process). אחת הבעיות הראשיות הנלוות לסקרים היא שמירה על עקביות ועל התמקדות בסוגיות החשובות ביותר. דרך פשוטה להשגת שתי מטרות אלו היא שימוש ברשימות תיוג מחושבות לכל פרטיהן, בהן יכולים הסוקרים להשתמש כסיוע להכנות לקראת פגישת הסקר. רשימות התיוג ממקדות את תשומת הלב בסוגיות החשובות ביותר ומסייעות לשמור על העקביות, משום שהן שואלות שאלות שמצביעות אל רמת הפירוט בה יש לעסוק בשעת ההכנה לפגישת הסקר. לרוע המזל, יש נטייה להזניח את רשימות התיוג, או לעשות בהן שימוש לא נכון. הסיבה לכך היא שהסוקרים נעשים בקיאים בהן יותר מדי, או שהרשימות עצמן חדלות לשקף שינויים שחלו במרוצת הזמן, וכך לפנינו קבוצה חדשה של 'בעיות חשובות ביותר'.

יומן השגיאות (error log) מסייע לזהות שינויים אלה, משום שהוא מראה איזה מהם נעשים שכיחים. כך הוא מספק עדות מוחשית למידת יעילות תהליך הסקירה. ניטור יומני השגיאות מספק התרעה מוקדמת לכל התדרדרות בתהליך הסקירה ומדרבן להכנסת שינויים ברשימות התיוג של הסקרים כדי שישקפו את הסוגיות הנוכחיות. בדוגמה שבתרשים 6.11, היינו מצפים שהרשימה שהניעה סקר זה תתייחס לשגיאות האופייניות לדיאגרמה של זרימת הנתונים, כגון, עקביות ודירוג לפי רמות כפי שנחשפו תוך כדי הסקירה. במקרה ולא, צריך לעדכן את הרשימות כדי שישקפו חולשה זו.

שנית, **התהליך כלפי מעלה** (upstream process). מה מספר לנו יומן השגיאות על התהליך כלפי מעלה? הוא יכול להוות מצביע אמין לגבי החולשות שבתהליך הפיתות, בכך שיראה באיזה מטכניקות התכנות יש סכנה גבוהה יותר ליצירת שגיאות. למידע זה נוכל להגיב על ידי מתן הנחיות או הדרכה למתכננים, על ידי עדכון הנהלים, או אפילו על ידי שינוי או החלפת הטכניקות בהן אנו משתמשים. בכל החלטה, יומן השגיאות יכול לשמש לניטור ההתקדמות, להבהיר לנו אם השגיאות צומצמו או חוסלו, או לא. בתרשים 6.11, השגיאות שבתרשים זרימת הנתונים נותנות שוב מקום לדאגה, ומצביעות על הצורך בהמשך מתן הדרכה כדי להזכיר למפתחים את הטכניקות שבשימוש בתהליך.

לבסוף, **התהליך כלפי מטה** (downstream process). במורד התהליך נמצאים כל שלבי הבדיקה שאנו עוסקים או אמורים לעסוק כעת בתכנונם. יומן השגיאות של סקר התיכון מציג בעיות שנתקלנו בתיכון, ומהו אופי השגיאות שעשינו עד עתה. מידע זה אנו יכולים לשקף כעת בתוכנית הבדיקות על ידי כך שנוודא שהתחומים החלשים שבתיכון ייבדקו ביעילות במהלך הבדיקות. עלינו גם לוודא שהבנו את ההשפעות האפשריות הרחבות יותר של השגיאות שגילינו בסקר, ושכללנו בדיקות שימנעו את התממשות שגיאות אלו. נראה שבתרשים 6.11 יש לנו חולשות אחדות במפרטי המודולים. כדאי יהיה להקדיש למודולים אלה תשומת לב מיוחדת בעת עריכת בדיקות היחידות הבודדות.

כיצד מתחילים בהכנת האסטרטגיה למדידה ולשיפור תהליכים

פרק זה הציג מספר נושאים ודרכים של המדידה ושל השימושים בה בשיפור התהליכים. באיזה מהם רצוי לך להשתמש בארגון שלך?

קל מאוד להתקע בפרטי תוכניות ובפרויקטי מדידה. יש צורך לעסוק כאן בכמות גדולה של פרטים חשובים, וכמעט וודאי שכל דבר שניתן למדידה יש לו ערך פוטנציאלי כלשהו עבורך. הבעיה היא שאין אפשרות להשתמש בו-זמנית בכל המידע הזה, שיש לו חשיבות פוטנציאלית.

מה שדרוש לך הוא אסטרטגיה; אסטרטגיה חייבת לכלול תמיד חוש כיוון.

וודא שהתהליכים יהיו נכונים

צעדי מפתח ביצירת אסטרטגיית מדידה

1. התחל בהחלטה לאן ברצונך להגיע. נסה לגלות את הבעיות הרציניות ביותר שעומדות בפני ההנהלה הבכירה (משום שהיא תצטרך לממן את תוכנית המדידה).

2. בחר בחמשת הנושאים הבעייתיים ביותר וטפל בהם לפי התור, התחל מהחמור ביותר.
3. טפל רק בבעיה אחת בזמן נתון.
4. אם הצלחת לקבל סמכויות לטפל בבעיה החשובה ביותר להנהלה שלך, הצלחת להתגבר על המכשול הראשון והקשה ביותר שלפניך.
5. תכנן ויישם בזהירות את אסטרטגיית המדידה שלך, ואז תהיה בעמדה חזקה בכל הנוגע להחזרת ערך ממשי למנהליך.
6. לכל אורך הדרך צפוי שתגלה סוגיות חדשות הטעונות טיפול. אל תרשה לעצמך לסטות מהמסלול. עם זאת, נהל רישום של סוגיות אלו והצג אותן כמטרות לפרויקטים עתידיים ככל שתגבר אמינותך.
7. תוכניות מדידה הולכות וצומחות. הנח לתוכניתך להתפתח בקצב טבעי, כדי שתוכל לצבור חוזק וחיות.
8. לא יהיה לך קשה להחליט מה יהיה נושא המדידה הבא. הבעיה שתעמוד לפניך תהיה, מה לדחות בעוד אתה עוסק בנושאים מעדיפות גבוהה יותר.

המדידה היא דיסציפלינה שהכל סובב סביב צירה: היא מאפשרת לכמת יעדי איכות; היא מאפשרת להשוות ביצועים מול יעדים; היא מאפשרת לגבש ולנסח יעדי שיפורים. ללא מדידה, האיכות יכולה להיות מושג מעורפל בלבד.

מה בדבר איכות המוצר?

בסופו של דבר, מה שחשוב לנו הוא איכות המוצרים שאנו מפיקים. תהליכים טובים יכולים להיות האמצעי הטוב ביותר להשגת איכות המוצר. עם זאת, התהליכים תמיד יהיו האמצעי ולא המטרה. מה שלא נעשה בתהליכים שלנו, עדיין אין בכך ערובה שיהיה בהם כדי לספק מוצרי איכות.

כיצד נוודא את איכות המוצר? לנושא זה שני אלמנטים. הראשון הוא **נוכחותן של תכונות איכות רצויות** כגון אמינות, ידידותיות למשתמש ויכולת של שימוש חוזר ברמות הנדרשות על ידי הלקוח. האלמנט השני הוא **השימושיות הכוללת של המוצר**. האם זהו באמת המוצר שהלקוח רוצה? כאשר אנו מעצבים מוצר לשיווק המוני, לא די בכך שהמוצר יענה על הדרישות הפונקציונליות שעל הנייר. מה שיקבע יותר מכל יהיה הערך הפנימי של המוצר עבור הלקוחות הפוטנציאלים שלו.

אין אסטרטגיה לשיפור תהליכים שיש בה כדי לקדם היבט זה של האיכות. יידרשו לכך צירוף של ניסיון, כישורים, חדשנות וידע שקשור קשר הדוק אל מחלקת הפיתוח המסוימת, אל השוק ואל סביבת הפיתוח שבה היא פועלת. כישורים אלה הם כישורי התיכון. בפרק הבא נדון באחדים מהיבטי התיכון.

חלק 4

ניהול איכות ושינויים טכנולוגיים

חלק 4 מציג נושאי תהליכי פיתוח תוכנה ומחזורי חיים 'לא-קוויים', בצורת פיתוח המסתמך על יצירת אבטיפוס ופיתוח מוכוון-עצמים.

◊ פרק 7: איכות ומחזורי חיים לא-קוויים.

◊ פרק 8: פיתוח יישומים מהיר.

◊ פרק 9: פיתוח מוכוון-עצמים.

מטרת חלק זה של הספר היא להעשיר את הבנתנו בכיוון חדש של טכנולוגיית התוכנה ולדון במשמעותו לגבי הגישה של מערכת ניהול איכות שהצגנו עד עתה.

בפרק 7 אנו מגדירים למה אנו מתכוונים בשם 'לא-קווי' ('non-linear') ומתווים את סוגי טכנולוגיות התוכנה המשתייכות לקבוצה זו. לראשונה אנו גם מנסים להגדיר את דרישות האיכות עבור פרויקטים לא-קוויים.

בפרק 8 נתבונן במיוחד בטכניקות המבוססות על יצירת אבטיפוס כגון 'פיתוח יישומים מהיר' (Rapid Applications Development - RAD) ו'פיתוח יישומים משותף' (Joint Applications Development - JAD), וננסה לזהות את משמעויותיהן לגבי ניהול איכות. סגנון פיתוח התפתחותי אבולוציוני מחייב אסטרטגיית פיתוח התואמת לטכניקה, ולכן אנו דנים באסטרטגיית איכות עבור RAD.

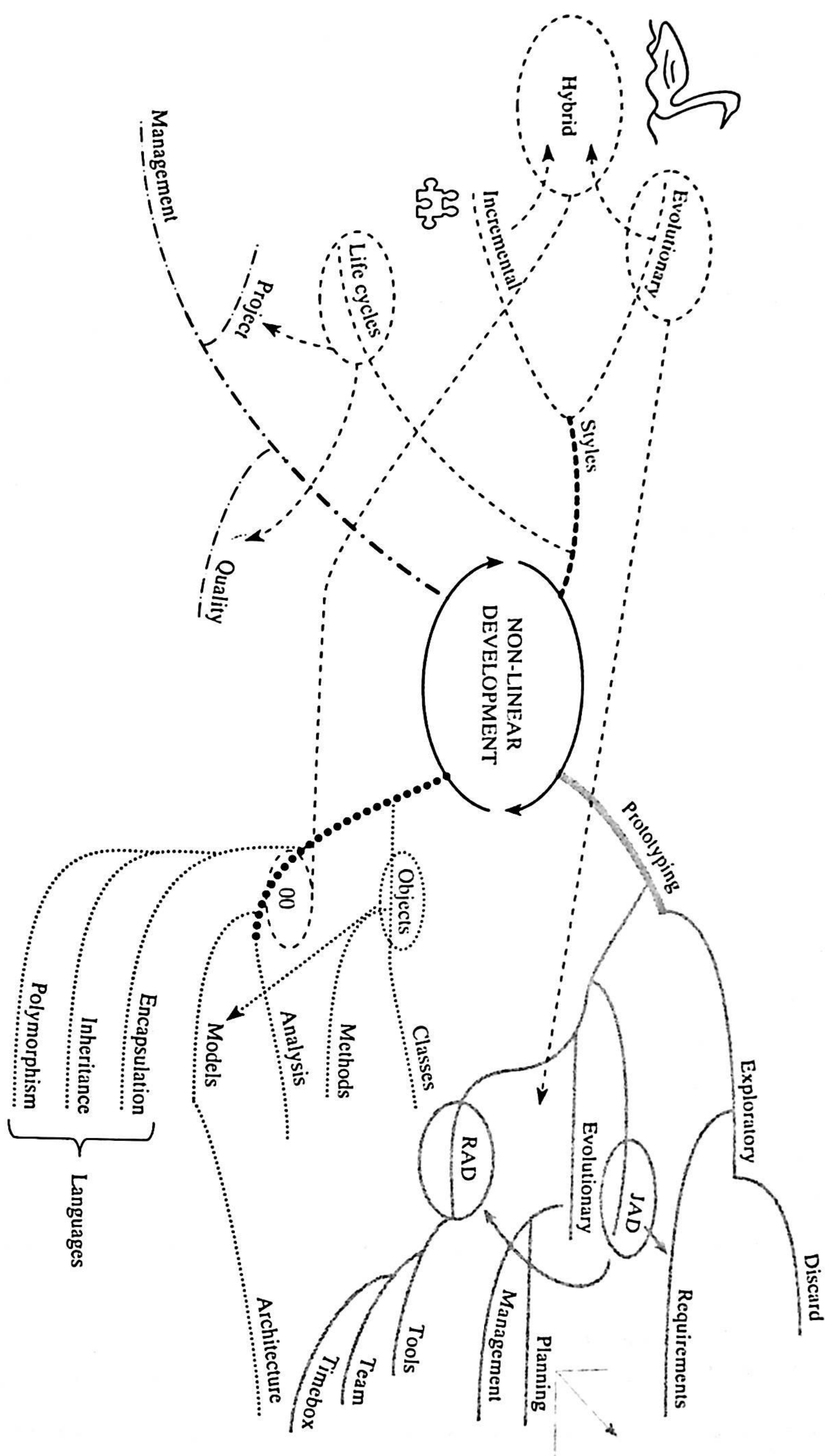
בפרק 9 אנו דנים בגישה מוכוונת-עצמים אל הפיתוח, ומוצאים שזוהי גישת כלאיים המורכבת משיטות תוספתיות ואבולוציוניות (incremental and evolutionary methods) כאחד. גם כאן נדון במשמעויות, ונתווה אסטרטגיה לטיפול בהן.

אחת ממטרות חלק זה של הספר היא להמחיש את העובדה ששיטות לא-קוויות מחייבות גישה גמישה, אך גם מבוקרת. תקן ISO 9000-3 נשאר תקף, אך כאן הוא מסוגל לסייע פחות מאשר בשיטות הקוויות. דרושות לנו הנחיות התואמות יותר לכיוון שלקראתו הטכנולוגיה מתקדמת. סיטואציה זו דומה למצב שיצר את תקן ISO 9000-3 - טיפול בשיטות 'לא-סטנדרטיות' הקשורות בטכנולוגיית פיתוח התוכנה.

בסוף חלק זה נוכל להבין את משמעויות האיכות של השימוש בשיטות לא-קוויות וכיצד לטפל בהן בפיתוח יישומים מהיר (RAD) ובפרויקטים מוכווני-עצמים.

תרשים ח.4.1. הוא מפת תפיסה המציגה את הקשרים בין רעיונות המפתח שבחלק.





תרשים ח.4.1 *מפת תפיסה המציגה את הקשרים בין רעיונות המפתח שבחלק*

פרק 7

איכות ומחזורי חיים לא-קוויים

מבוא

מאז ומתמיד הופעל לחץ על שיטות פיתוח התוכנה להגיע לפריון גבוה יותר, ותמיד בהנחה שפירוש הדבר יותר קוד בפחות זמן ומשאבים. תביעות אלו לפריון גבוה יותר וללוחות זמנים מקוצרים מוליכות את מפתחי התוכנה לאמץ גישות איטרטיביות והתפתחותיות, בהן אחת המטרות היא למזער את תקורת התיעוד ולהגביר את המאמץ המוקדש לתיכון. האם שיטות לא-קוויות אלו הן בטוחות? האם פקעה זכות הקיום של השיטות המובנות? האם פירוש המגמה הנוכחית הוא, שהענף המתבגר אינו זקוק יותר לתמיכת תהליכי התיכון המובנה, או שפירוש הדבר כניעה סופית לתוהו ובוהו אשר איים תמיד לשטוף את קהיליית התוכנה? איך שלא נתבונן במצב, בעתיד, המעבר לשיטות אפקטיביות לניהול האיכות בסביבה זו, יהיה קריטי להצלחה, ואפילו להישרדות.

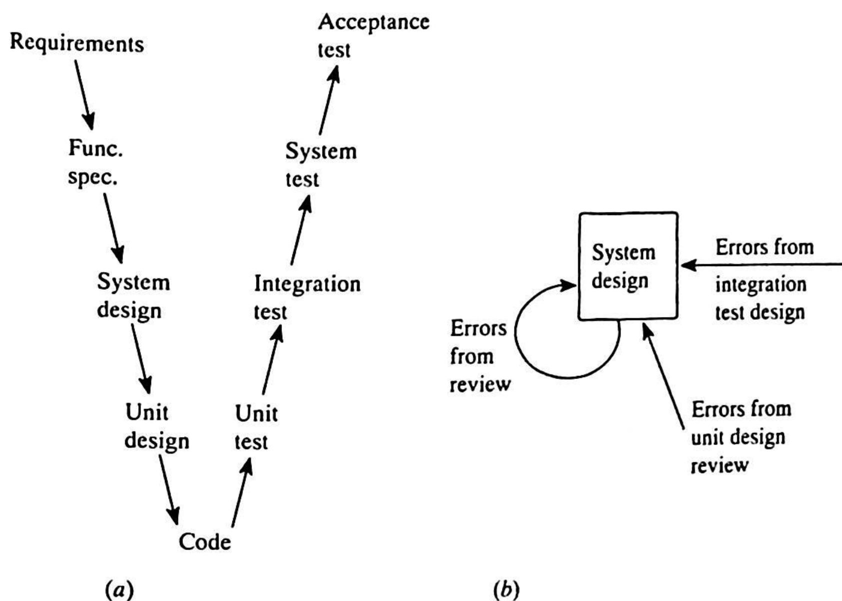
מחזורי חיים קוויים ולא-קוויים

מהו מחזור חיים? איזו מטרה הוא משרת? את כל מה שאמרנו עד עתה קישרנו למודל של **מחזור חיים** (life cycle). הדבר מצביע בבהירות על החשיבות שאנו מייחסים לרעיון של מודל מחזור החיים כדיסציפלינה יסודית וכקו-בסיס עבור כל הפרטים של שיטות, כלים וטכניקות.

מחזורי החיים בהם השתמשנו עד כה היו קוויים כולם. במונח *'קווי'* (linear) כוונתנו לרצף מוגדר שמתחיל מהדרישות, דרך התיכון והיישום, אל מסירתו הסופית של המוצר. בהכרח, חייב הרצף לכלול מידה מסוימת של איטרציה, כמו למשל, לצורך תיקון שגיאות, או לצורך הכללת שינויים מבוקשים; אך האיטרציה עצמה משנית לרצף הפעילויות.

במחזור חיים קווי, הרצף המוגדר מאפשר לנו לבנות תוכניות מפורטות של תהליך הפיתוח, ולייחס תוכניות ואבני דרך לרמות המתפתחות של פירוט התיכון. החשיבה הקווית היא תהליך טבעי לנו. היא מייצרת תחושה של *'קונקרטיות'*, כלומר גישה מעשית ועניינית. היא גם מטפחת את הביטחון החיוני ביכולתנו לספק מוצר איכות שעונה על הדרישות. תרשים 7.1 מדגים את האופי הרציף של מחזור חיים מסוג V,

למרות שהחזרה ממלאת בו תפקיד ראשי משום שהיא מוודאת את היכולת להגיע לפתרון שלם ונכון. במחזור חיים קווי, האיטרציה ממלאת תפקיד חשוב באימות ובבדיקת התקפות (validation); יש לה השפעה נכבדה על הערך והאיכות של מוצר, למרות העובדה שהיא יוצרת קשיים רציניים בתכנון ובבקרה.



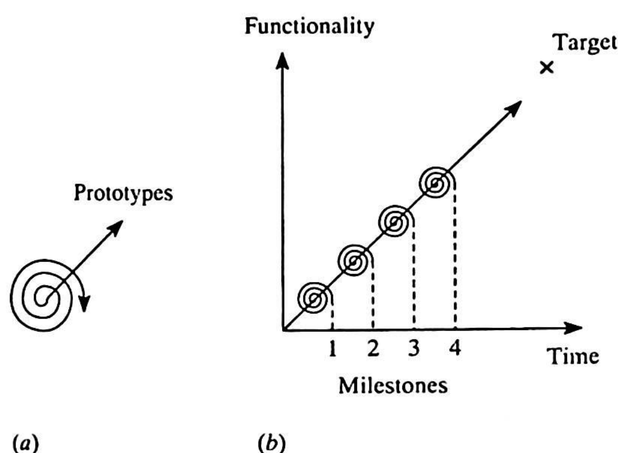
תרשים 7.1 מחזור חיים רציף: (a) הרצף (b) איטרציה של שלב

ההתעקשות להיצמד לתוכנית ערוכה ברצף, אחראית לכשלונות רבים בזמן מבחני הקבלה; שגיאות שהוסתרו במשך תקופה ממושכת צפות ועולות על פני השטח. כמו כן, הפיתוי לשוב ולחזור על תהליכים עד לרגע בו תושג סוף סוף השלמות, אחראי אף הוא למספר רב של פרויקטים שיוצאים משליטה ובסופו של דבר נזנחים. הסיבות להזנה שונות: אם משום שהפרויקטים יקרים מדי, או משום שהם הופכים ל'זוללי משאבים'. איזון בין שני קצוות אלה מושג בתוכנית האיכות. זהו המקום בו מנהל פרויקט זהיר משליט מדיניות ברורה המאפשרת קבלת הערכה נאותה של הסיכונים והשגיאות בכל אחד מהשלבים, מבלי לעודד שיתוק היכול להתפתח כתוצאה מגישה קנאית מדי לאיכות.

מה קורה כאשר מחזור החיים הוא '**לא-קווי**' (non-linear)? כאשר אנו אומרים לא-קווי כוונתנו שהתהליך הראשי הוא איטרטיבי או התפתחותי, כרוך בפיתוח פונקציונליות באמצעות מודל או אבטיפוס, אשר מתקרבים אל הפתרון הנדרש תוך סדרה של שיפורים ועידונים. תרשים 7.2 מציג את האופי האיטרטיבי של מחזור החיים החלזוני (הספיראלי), אף שבדרך כלל תידרש סדרה של מודלים כדי להגיע לפתרון קביל. במחזור חיים לא-קווי, התהליך האיטרטיבי הבסיסי פועל כספק הראשוני של ערך ואיכות, ואילו הרצף נדרש כדי לוודא שתהליך בניית המודל מתקרב לפתרון קביל. הרצף הוא חלק מהדרך בה מוודאים את ההתקדמות העקבית לקראת מטרות ויעדים

מוגדרים, אך בנקודת המוצא, מספר השלבים בו ואורך כל שלב אינם ידועים עדיין. מטבע הדברים, התכנון והבקרה קשים יותר מאשר במחזור חיים קווי.

במקרה של מחזור חיים לא-קווי, תשומת לב רבה מדי שניתנת לאיטרציות על חשבון הרצף, עלולה להוביל לפרויקטים לא ממוקדים. עם זאת, ייתכן שמהבחינה הפוטנציאלית תושגנה תוצאות בעלות ערך רב, כתוצאה מתרגילי אבטיפוס. לעיתים תכופות קורה שפרויקטים נזנחים משום שהעלות והמאמץ הכרוכים בהפקת תוצאות שימושיות כלשהן עוברים כל גבול. מאידך, ניסיון המבקש להתקרב מוקדם מדי למודל יסופי, מוליך לעיתים תכופות לפתרונות מורכבים ולא גמישים. גם כאן, תוכנית האיכות היא המקום המתאים בו יש להשליט איזון בין האיטרציה לבין הרצף. עם זאת, האם יש ביטחון שדיסציפלינות האיכות הפועלות בהצלחה במחזור חיים קווי, תפעלנה בהצלחה גם בתהליך האיטרטיבי או ההתפתחותי?



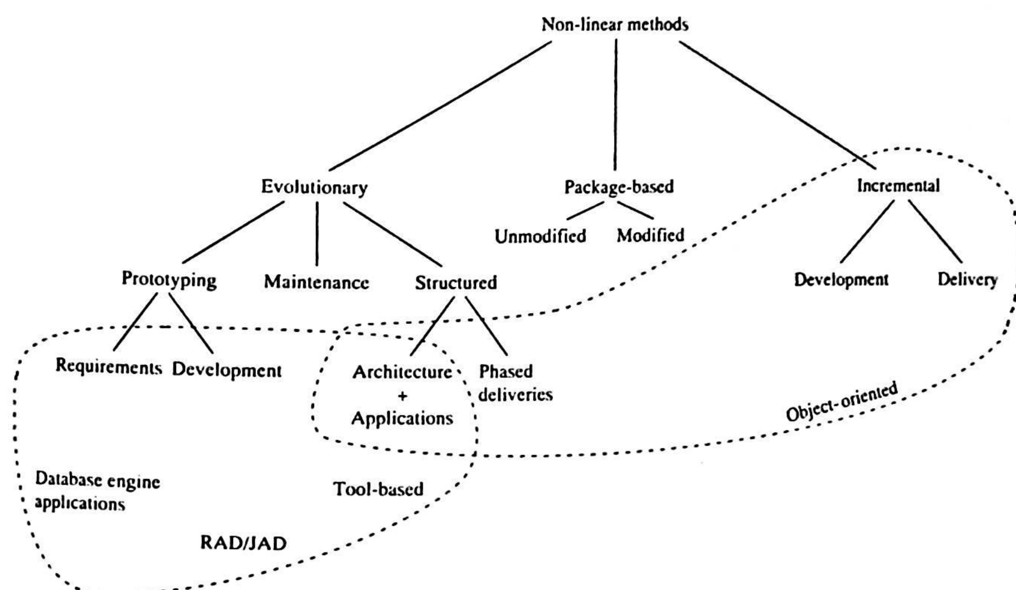
***תרשים 7.2** מחזור חיים איטרטיבי: (a) הספירלה האיטרטיבית הבסיסית (b) רצף של ספירלות*

מהן השיטות הלא-קוויות

לצורך הדיון שלנו נגדיר את השיטות הלא-קוויות כשיטות בהן האיטרציה ממלאת תפקיד משמעותי יותר מאשר בשיטות הרצף המקובלות. **נכלול גישות תוספתיות והתפתחותיות** (incremental and evolutionary methods), פיתוחים מבוססי-חבילת-תוכנה ותחזוקה שגרתית. בתרשים 7.3 מוצעת דרך למיון השיטות הלא-קוויות.

באווירה הכוללת של שיטות לא-קוויות, מתבלטות שתי טכניקות כבעלות משמעות מיוחדת: פיתוח מהיר של יישומים, ופיתוח מוכוון-עצמים. בטכניקות אלו ניתן להשתמש בכל אחת מהגישות הלא-קוויות (בפיתוח מוכוון-עצמים משתמשים גם במסגרת הקווית). בשני הפרקים הבאים נעסוק בטכניקות אלו, בשל המשמעות וההשפעה הרבה שיש להן על האיכות.

בשאר חלקי פרק זה, נבחן את אופי השיטות הלא-קוויות, ונתבונן בדרכים בהן יכולים לסייע לנו הקווים המנחים של תקן ISO 9000-3 לשימוש בשיטות אלו.



תרשים 7.3 דרך למיון שיטות לא-קוויות

התפתחות התוכנה

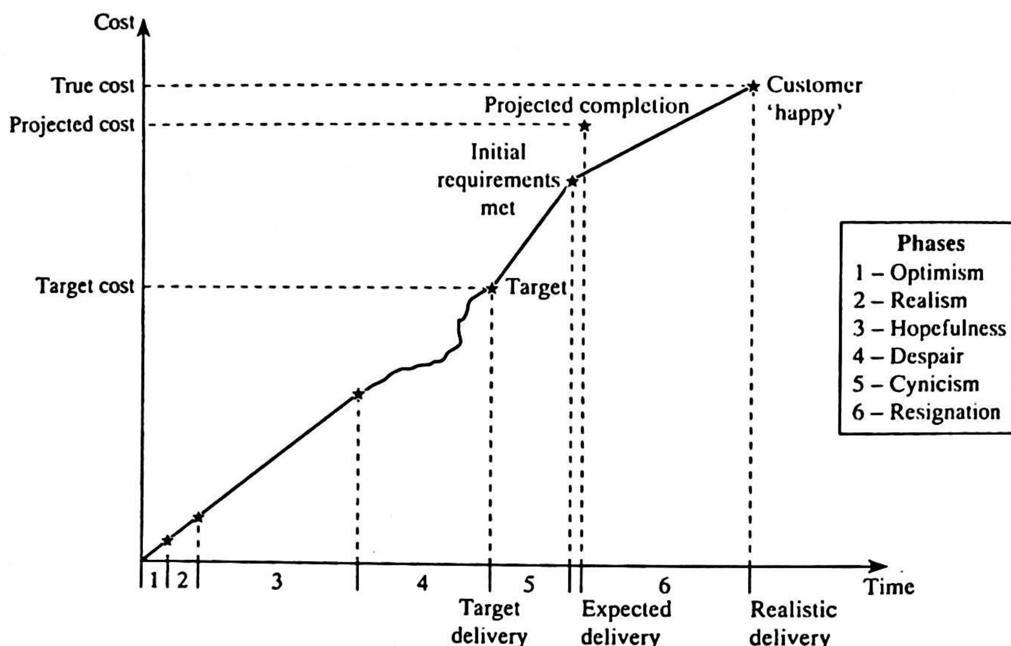
התפתחות מתוכננת ולא מתוכננת

כל תוכנה נוצרת, בין אם כתוצאה מתוכנית מחושבת מראש, ובין אם כתוצאה מאי יכולת להשיג את היעדים הרצויים בניסיון הראשון. פרויקטים יומרניים רבים סובלים מתוצאות לא-נמנעות של הניסיון לדחוס כמות פיתוח מרובה ללוח זמנים קבוע מראש.

בפרק 3 בחנו תרחיש קלאסי, בו חקרנו את ההשפעות האפשריות של אילוץ פרויקט פיתוח, שלא הוגדר כהלכה, לתוך תקציב ולוח זמנים ספציפיים. העתקנו את תרשים 3.1 כתרשים 7.4, כדי להזכיר לך את הבעיות בהן נתקלנו.

שם הגענו למסקנה שגם אם הפרויקט מסתיים בזמן, יש מקום לצפות שהוא יכיל פערים וחולשות נכבדים. איכות התוכנה לא איפשרה עבודה מחדש (rework), ולמעשה, ביצוע חוזר של פעילויות שהיינו נוקטים כדי לנסות להביא לכך שלכל הפחות ניתן יהיה להשתמש במערכת. בדרך כלל, ההיבטים רבים ולא-צפויים של המערכת היו

מביאים לעבודה חוזרת. בסופו של דבר, התוכנה יכולה היתה להימסר במצב שמיש, אך 'רגישי' במקצת. בשלב זה נוצר צורך לטפל, במסגרת שלב התחזוקה, באוסף הגדול של שינויים מבוקשים (שיהוקפאו' כדי לאפשר את השלמת העבודה מחדש).



תרשים 7.4 *תהליך פיתוח סכימתי עם מגבלות שנקבעו*

במסגרת זאת, עבודה מחדש מייצגת חלק גדול ביותר של כלל השקעת המשאבים בפרויקט. אין זה בלתי שכיח שבתנאים מסוימים, מגיעה עלות העיבוד החוזר עד כדי 50 אחוז מהעלות הכוללת של הפרויקט. בשל אופי העבודה הזו, מעטים הסיכויים שנוכל לתכננה היטב. בהיותה מונעת על ידי דוחות של שגיאות ועל ידי דרישות לשינויים, רבים הסיכויים שפאזת העיבוד החוזר תתנהל כתרגיל לא מתוכנן ביצירת אבטיפוסים, ותסתיים כתוצאה מלחץ גובר והולך לספק את המערכת גם אם הפתרונות שנעזרו באבטיפוסים לא עובדו במלואם. שלב התחזוקה הוא תרגיל נוסף ודומה בתרגיל ההתפתחותי, אלא שהוא משתרע על פני לוח זמנים ארוך יותר ומבוסס על הבסיס הלא-יציב של התוכנה, שהועבר לשלב התחזוקה בגמר העבודה מחדש.

פרויקט זה, שתוכנן כפיתוח רציף יחיד, נמסר בסופו של דבר כהתפתחות תלת-שלבית. בכל הפרויקטים, בין אם הם מבוססים על שיטות קוויות או לא-קוויות, כרוכים שני שלבי התפתחות: שלב הפיתוח ושלב העבודה החוזרת. מעטים הפרויקטים המתוכננים כפיתוחים התפתחותיים.

ראוי היה שהפרויקט שלנו יתוכנן בשלבים, כשכל שלב נבחר על סמך העדיפויות של המשתמשים ועל הערכה מציאותית של הניתן להשגה. בדרך זו, הסיכויים להשגת תוצאה מוצלחת עבור המשתמשים, היו עולים בצורה דרמטית. בנוסף, ניתן היה למנוע את ההשפעות השליליות על עיצוב התוכנה. כתוצאה מכך, המוצר היה במצב בו הפיתוח בטווח הארוך, היה נעשה קשה פחות.

יתרונות החשיבה ההתפתחותית גלויים לעין, אלא שיש גם צורך לשכנע את המשתמשים שלהתפתחות יש גם יתרונות עבורם. ככלות הכל, ככל שנראה לעין, הגישה ההתפתחותית כרוכה באיחור באספקה של חלק מדרישות המשתמשים.

נקודה חיונית שיש לעכלה כאן היא, שהגישה הקווית לא תמיד מספקת את כל הדרישות בזמן. רוב הארגונים המזמינים פרויקטים גדולים, שנמשכים למעלה מ- 12 חודשים, למשל, מגלים שבעיות מהסוג שתיארנו שכיחות למדי. כלומר, **התאריך שנמסר בפועל כתאריך היעד, לא רק שאינו שלם, אלא גם ייתכן שלא ישקף את הדרישות הדחופות ביותר של המשתמשים.** כדי להתגבר על בעיה זו, עלינו לתכנן בכיוון גישת אספקה התפתחותית המשקפת את דרישות המשתמשים.

יותר ויותר הולך ומתברר שמועטים סיכוייהם של פרויקטים בעלי אופק תכנון העולה על שישה חודשים, להצליח. את הסיבה לכך ניתן לייחס רק בחלקה לעובדה שהמפתחים מתקשים לתכנן, או ליישם פרויקטים מסדר גודל זה. עוד יותר חשובה העובדה שהמשתמשים מתקשים לקבל על עצמם מחויבות מלאה לפרויקטים החורגים בהרבה מאופק התכנון האישי שלהם. רובנו מרגישים לא בנוח עם הרעיון של התבוננות בעתיד מרוחק מדי; זה נכון במיוחד כאשר אנו מתכננים שינוי משמעותי. אנו יכולים להקיף במחשבתנו יעדים ארוכי טווח, ואפילו חלומות לטווח ארוך עוד יותר, אך לא תמיד אנו יכולים להמיר את כוונותינו לתוכניות מגובשות. הניסיון מלמד, שיכולתנו לדמיין בדיוק תוצאות של תוכנית מפורטת, מצטמצמת לטווח של שישה חודשים. לכן, תוכניות המשתרעות אל מעבר לשישה חודשים קדימה, נתונות לסיכונים כבדים. אינסטינקטיבית, כבר מלכתחילה איננו נותנים אמון רב בתוכניות המתרחבות מעבר לאופק זה. כתוצאה, המשתמשים מציבים את ההשתתפות בפרויקטים ארוכי טווח במקום נמוך בסולם העדיפויות שלהם. הם דוחים זאת למועד בו נמצא הפרויקט בתוך ששת החודשים שלפני מועד האספקה. במועד זה כבר מאוחר מדי לתרום תרומה שימושית כלשהי, מבלי לגרום להפרעות רציניות.

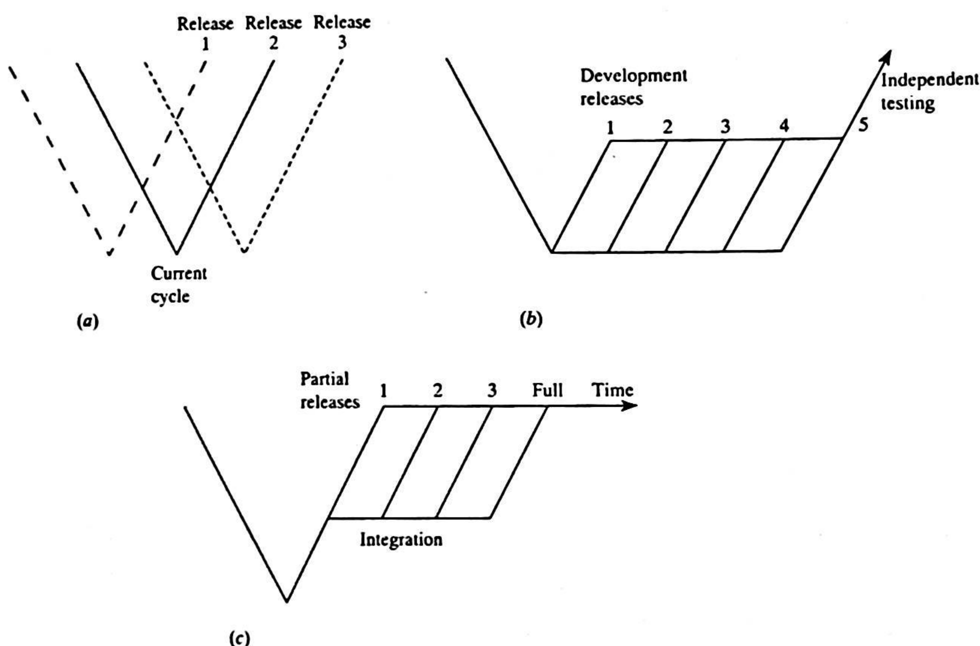
שיטות התפתחותיות

ההחלטה על נקיטת ***גישה התפתחותית*** (evolutionary approach) לפיתוח, משאירה פתוחה את השאלה של דרך הגישה אל תרגיל הפיתוח. עד עתה נסב הדיון על שיטות פיתוח מסורתיות המלוות בגישה התפתחותית אל התכנון. עם זאת, יש שיטות אחרות התואמות טוב יותר לסגנון ההתפתחותי. שיטות שמשתמשות ביצירת אבטיפוס יידונו בפרק 8, ובפרק 9 נעסוק בשיטות המשתמשות בגישה מוכוונת-העצמים.

פיתוח תוספתי

לצורך דיון זה נגדיר את **הפיתוח התוספתי** (incremental development) כגישה הכרוכה בפיתוח או באספקת מערכות, קטע אחר קטע. כל תוספת מכילה חלק מהפונקציונליות של המערכת הכוללת. ניתן להשוות את התהליך להרכבת חלקים של חידת פאזל, בה כל חלק מכיל קטע תמונה חיוני, אך נעזר בשכניו להשלמת הפאזל.

סגנון הפיתוח התוספתי כרוך במספר איטרציות גבוה יותר מזה המקובל בפיתוח הקווי המסורתי. עם זאת, ניתן עדיין לנהל את הפיתוח התוספתי בתוך מסגרת פרויקט קווי. תרשים 7.5 מציג אחדות מהצורות הרבות של פיתוח תוספתי המבוססות על מחזור החיים בסגנון V המוכר לנו.



תרשים 7.5 גישות של פיתוח תוספתי: (a) גרסאות בריבוי מערכות; (b) גרסאות בריבוי פיתוחים; (c) הספקה בריבוי תוספות

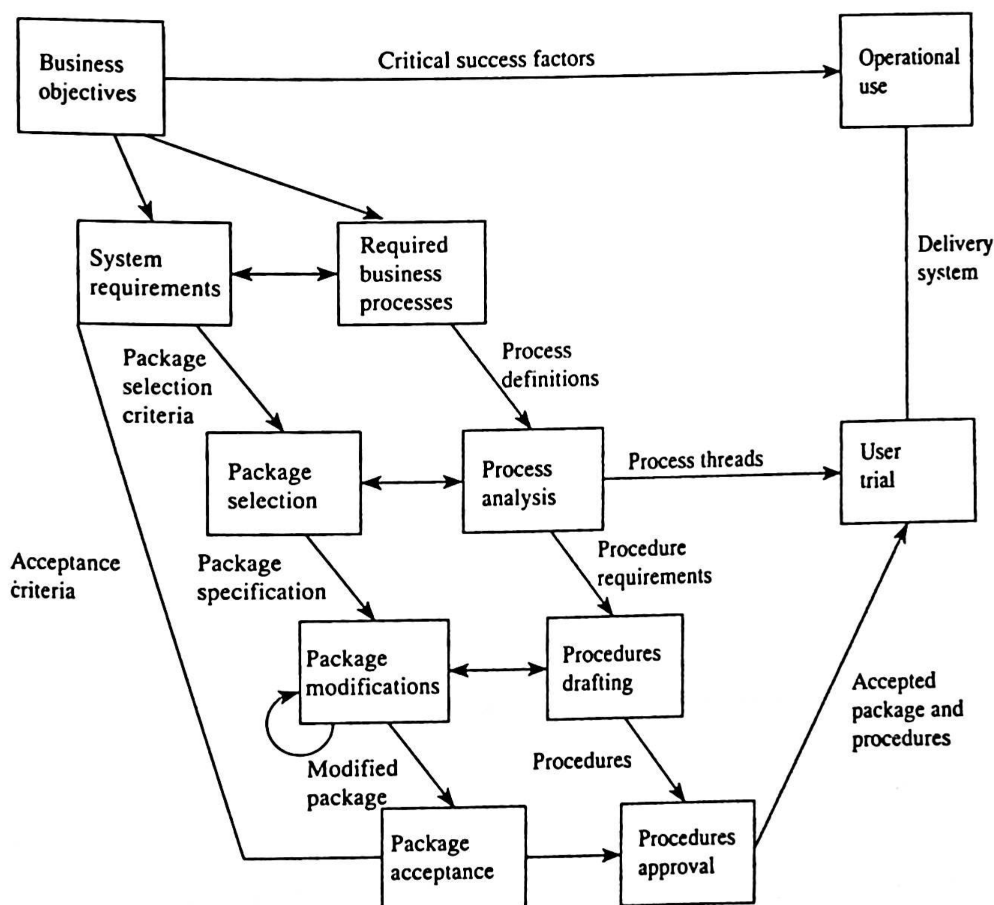
תרשים 7.5(a) מציג **ריבוי גרסאות מערכת** (multiple system releases) המבוסס על פיתוח מקביל של קטלוג דרישות, מפוצל למחיצות. במקרה זה, תוכלנה התוספות להתבסס על **קווי קישור פונקציונליים** (functional threads), על יישומים הניגשים אל אזורים שונים בבסיס הנתונים, או על כל מחיצה שמספקת מידה מסוימת של אי-תלות בין המחיצות, הדבר מאפשר לפתח אותן במקביל. צפוי שיהיה מספר גבוה של איטרציות כאשר משלבים בתוך ההספקה הבאה את הבעיות ואת השינויים שנדרשו בעת ההספקה הקודמת.

תרשים 7.5(b) מציג **ריבוי גרסאות פיתוח** (multiple development releases), המועברות מהפיתוח אל ניסויי פיתוח עצמאיים. התיכון והקידוד המפורטים צפויים כאן לאיטרציות בין הגרסאות השונות של ניסויי הפיתוח.

תרשים 7.5(c) מציג **ריבוי הספקות** (multiple incremental deliveries) מתיכון יחיד. במקרה זה מחולקים שלבי היישום והבדיקות, כך שניתן לבצע הספקה בחלקים, כתוספת. כך עולה אמון הלקוח, וכך גם מתאפשרת עליה באיכות ההספקות התוספתיות, כתוצאה מתיקון שגיאות בשלב מוקדם של מחזור האספקה. בדרך כלל, התוספות מבוססות על קווי קישור פונקציונליים ומסופקות לפי סדר העדיפויות.

למודלים פשוטים אלה חלופות רבות, אך כולן מתאפיינות ברכיב איטרטיבי המשולב בתוך מחזור חיים קווי. פיתוח תוספתי הוא למעשה, הכלאה של שיטות קוויות ולא-קוויות, ויש להתייחס אליו בהתאם. התכנון לקראת פרויקט תוספתי צריך להיות מבוסס על מודל קווי. עם זאת, יש להיזהר בתכנון השלבים האיטרטיביים. שלא כמו הפיתוח ההתפתחותי שאינו מוגבל בזמן, השלב האיטרטיבי של הפיתוח התוספתי חייב לספק את המוצר במסגרת לוח זמנים נתון. התוכנית חייבת לזהות אבני דרך בסוף השלבים האיטרטיביים, שתציינה את המועד שבו השלב האיטרטיבי חייב להסתיים.

שיטות מבוססות-חבילת-תוכנה



תרשים 7.6 מחזור חיים של פיתוח מבוסס-חבילת-תוכנה

הספקת יישומים המבוססת על חבילות תוכנה סטנדרטיות היא מקרה פרטי של הפיתוח הלא-קווי. אין זה משנה אם צריך להכניס שינויים בחבילה. בכל מקרה חייב להיות שלב בפרויקט בו יתכונן הארגון לקראת התקנת החבילה, למשל על ידי כתיבת נהלים חדשים, ויתכונן לקראת השלב בו תיושם החבילה בפועל. פעילויות אלו דומות ליצירת אבטיפוס, אלא שהן מיושמות לגבי הארגון, ולא לגבי יישום התוכנה בדרך כלל, הן כוללות הדמיה של סביבת היישום החדשה.

בכל מקרה בו צריך לשנות חבילה סטנדרטית לפני יישומה, ידמה שלב הפיתוח לשלבים מאוחרים יותר של הפיתוח ההתפתחותי. אלא שכאן יש צורך גם בשלב של ניתוח הדרישות כדי לוודא שהשינויים הוגדרו במלואם ובצורה הנכונה. מחזור החיים המתקבל ידמה למחזור V קטום, עבור האספקה הראשונה. לאחר מכן יחזור להיות מחזור חיים התפתחותי, שעה שבמקביל נעשים בארגון שינויים בנהלים ובתוכנה, כדי להגיע לשילוב מתקבל על הדעת.

תרשים 7.6 מציג את מחזור החיים של פיתוח מבוסס-חבילה.

התחזוקה כתהליך התפתחותי

תחזוקת מוצר תוכנה כלשהו, ואין זה משנה כיצד פיתחו אותו, כרוכה בהכנסת שינויים בתיכון קיים. התחזוקה יכולה להיות כרוכה בתיקון שגיאות, הרחבה או העשרה של יישום, ותיתכן אף התקנה של מספר שינויים בו-זמנית. קיים סיכון שההתפתחות תתנהל בכיוון לא רצוי, או ששינויים בתחום אחד יפגעו במשתמשים בתחום אחר, אלא אם כן כל השינויים יתוכננו ויוערכו מראש.

התפתחות (אבולוציה) באמצעות תחזוקה, מחייבת תוכנית התפתחותית מבוססת על קבוצת יעדים עסקיים מתאימים. שינויים מוצעים התואמים למטרות אלו יכולים לקבל אישור במינימום השהייה. עם זאת, אישור זה יהיה כפוף לאישור כל קבוצות המשתמשים המעוניינות. שינויים שאינם כלולים בתחום היעדים המוגדרים, טעונים שיקול דעת זהיר. אישור שינויים אלה משמעותו שינויים מסוימים ביעדים הכלליים, ולכן הוא חייב להישקל על ידי הרשות המתאימה. בכל מקרה, יש צורך לסקור את היעדים הכלליים אחת לשנה.

הנחיות תקן ISO 9000-3 מיושמות לשיטות לא-קוויות

האם תקן ISO 9000-3 ישים גם לגבי שיטות לא-קוויות? תקן זה אינו מכתיב את השימוש במחזור חיים קווי, אך הנחיית התקן מוצגת בצורה קווית, ומבנה המסמך תואם להפליא את מחזור החיים שבסגנון V. מובן שאין משתמע מכך שיש להעדיף את מחזור החיים בסגנון V. עם זאת, יהיה עלינו לעמול קשה כדי להתאים חלק מהנחיות תקן ISO 9000-3, לשיטה לא-קווית.

תקן ISO 9000-3 ומחזורי חיים לא-קוויים

שיטות לא-קוויות מציבות מספר בעיות בבואנו לפרש את הנחיות תקן ISO 9000-3. היכן מתעוררות הבעיות? בעיקר בקטע הדן בפעילויות של מחזור החיים (המספרים שבסוגריים מתייחסים לסעיפי התקן, ראה פרק 2):

- תכנון הפיתוח (5.4), ובמיוחד:
  - ◊ הגדרת השלבים;
  - ◊ לוח זמנים של הפרויקט, כולל זיהוי כל המשימות לביצוע.
- שלבים (5.4.2.1), במיוחד קלט ופלט של השלבים.
- הנהלה (5.4.2.2), במיוחד לוחות זמנים.
- בקרת התקדמות (5.4.3).
- דרישות לשלבי הפיתוח (5.4.4).
- תוצר שלבי הפיתוח (5.4.5).
- אימות כל שלב (5.4.6).
- תכנון האיכות (5.5).
- תָּכֶן ויישום (5.6), ובמיוחד סקרים (5.6.4).
- תכנון הבדיקה (5.7.2), ובמיוחד:
  - ◊ תוכניות עבור פריט תוכנה, שילוב, ניסויי מערכת, ובדיקות קבלה;
  - ◊ עיצוב בדיקות עבור בדיקות ביניים;
- בדיקות (5.7.3), ובמיוחד בדיקות רגרסיה.

כל התחומים האלה מחייבים פרשנות מדוקדקת. אחרים, כגון ניהול התצורה, טעונים תשומת לב מיוחדת בעת היישום.

ניהול פרויקט - הגדרת שלבי הפרויקט

ניהול הפרויקט מטופל ראשון משום שזהו תחום הסיכון הבולט ביותר. אחד המניעים העיקריים להגדרת מודלים של מחזורי חיים קוויים, הוא הצורך לספק למנהל הפרויקט מסגרת לתכנון ולבקרה של הפרויקט. המודל הלא-קווי, שהוא יחסית **חסר תכונות** (featureless) או מאפיינים, כמעט ואינו מספק נקודות אחיזה יציבות; אבני הדרך היחידות שמוגדרות הן התחלה, וסיום מחזור המידול.

מהו 'שלבי' בשיטה לא-קווית? המועמד הבולט לתואר זה הוא מחזור המידול היחיד, אך השיטות המעשיות נוטות לעשות את פיתוח המודל לתהליך מתמשך; המודל מתפתח כל הזמן. כדי לזהות 'שלבים' באותה משמעות שתקן ISO 9000-3 מגדיר אותה (סעיפים 3.5 ו- 5.4.2.1), נצטרך להרכיב איזה שהוא מבנה-על מעל לתהליך ההתפתחותי. מבנה העל מספק אלמנט קווי, בכך שהוא ממיר את הספירלה לסדרת ספירלות קטנות יותר, כפי שמוצג בתרשים 7.2(b).

כפייה זו של הרצף על התהליך האיטרטיבי, יוצרת מבנה בו נוכל לדמיין לעצמנו בדרך קלה יותר את התכנון והבקרה. עם זאת, אסור להניח לפונקציות התכנון והבקרה להשתלט על התהליך האיטרטיבי, ובכך לגרוע מיתרונות ההתפתחות השלבית (איטרטיבית). כל ספירלה ממשיכה להיות איטרטיבית, אך התהליך בכללותו מפוצל למספר צעדים התפתחותיים. ניתן לתכנן כל צעד התפתחותי כחלק מהפיתוח הכולל, ולהופכו בכך לישלבי. כל ישלבי בפני עצמו, ניתן לחלוקת משנה נוספת במקרה הצורך, ואפשר לכוונו אל יעד ספציפי, שהוא כשלעצמו רכיב של יעדי הפרויקט הכוללים.

ניטור ההתקדמות

ניטור ההתקדמות מנצל את מבנה התכנון כמסגרת בסיסית לזיהוי ולמדידת ההתקדמות. במקרה הלא-קווי, מבנה התכנון מגושם יחסית ובדרך כלל (במקרה הטוב), מספק רק את הרצף ההתחלתי של מחזורי המידול, בהתבסס על ניתוח כולל של היעדים העסקיים.

תוכנית השלב מכילה פירוט מספיק המאפשר לעקוב ולדעת אם מחזורי המידול המתוכננים הסתיימו אם לאו. עם זאת, אין בפירוט זה כדי לדעת מהי ההתקדמות בפועל, בכיוון השגת היעדים העסקיים. ייתכן שנמצא שאנו עומדים בקצב, בכל הנוגע לתוכנית הפיתוח, אך המודלים שתוכננו בתחילה אינם מתאימים יותר להשגת יעדי העסק. תקן ISO 9000-3 מצפה שסקרי ההתקדמות יוודאו את הביצוע האפקטיבי של תוכניות הפיתוח (5.4.3). משמעות הדבר, עיון חוזר בתוכנית עצמה ועדכונה בהתאם לצורך, כדי לוודא שהיא משקפת בצורה נכונה את יעדי העסק. עדכון התוכנית מטופל בעיקר בסעיף 5.4.1.

דרושה מידה מסוימת של הערכת כל מודל בסוף מחזור הפיתוח, כדי לקבוע אם המודל המסוים השיג את מטרותיו, ואם הפרויקט הכולל יכול עדיין להשיג את היעדים הכוללים. ניטור ההתקדמות חייב להיות יותר מאשר תיוג סתם, לציון עובדת הסיום של שלב בפרויקט. עליו לאפשר לנו להגיע לכלל הבנה אמיתית לגבי *המקום שבו נמצא הפרויקט*.

עלינו לדעת את הדברים הבאים:

- האם הפרויקט מסוגל עדיין לעמוד ביעדים;
- במקרה שלא, מה הם השינויים שיש להכניס ביעדים ההתחלתיים;
- מתי יש סיכויים להשגת היעדים;
- העלות של השלמת הפרויקט;
- האם הפרויקט ממשיך להיות בעל זכות קיום.

המידע הדרוש לנו צריך להיות איכותי וכמותי גם יחד, ובהשגתו חייבים להיות מעורבים גם נותן החסות לפרויקט וגם צוות הפיתוח.

הבנת מצבו האמיתי של הפרויקט, מעצם טבעה, קשה יותר מאשר במקרה של תהליך קווי מובנה. היא נסמכת במידה רבה, על התעניינות מתמשכת ומעורבות נותן החסות

העסקי. הוא צריך להיות מסוגל לשייך את יעדי כל אחד משלבי המידול למקומו, כחלק מהיעדים הכלליים, ולהגיע למסקנות על מצבו הכללי של הפרויקט.

תכנון ותכנון חוזר

תהליך לא-קווי צריך להיות בעל שתי שכבות תכנון:

- תכנון כולל של הפרויקט ברמת המאקרו, בצורת רצף שלבי מידול;
- תוכנית ברמת המיקרו, עבור כל אחד משלבי המידול.

חיוני שהתוכנית ברמת המאקרו תיווצר מתוך היעדים הכוללים, עוד לפני התחלת העבודה. תוכניות ברמת המיקרו, לעומת זאת, תהיינה מותנות בתוצאה של הערכת המודל הקודם. תכנון ותכנון חוזר הם במהותם פעילויות דינמיות, ומהווים במידה רבה חלק מתהליך הפיתוח עצמו.

המודל הקווי של תהליך הפיתוח, יחד עם תהליך ניהול פרויקט, שהוא מקביל וגם עצמאי במידה רבה, פותחים פתח למודל של שני תהליכים הקשורים ושזורים זה בזה, ומותנים במידה מכרעת זה בזה בכל צעד.

תקן ISO 9000-3 מתמקד בתוכנית פיתוח, המתעדכנת באופן סדיר. לעומתה נמדדת דרך קבע ההתקדמות שמתאימה למודל הפיתוח הקווי, ועם זאת, אינה מותחת את תכונות המפתח של תהליך הפיתוח הלא-קווי.

תכנון איכות עבור פיתוח לא-קווי

תכנון איכות עוסק בקביעת יעדי איכות, ובהגדרת תהליכי ניהול מתאימים כדי לוודא את השגתם (ISO 9000-3 סעיף 5.5). כיצד משפיע תהליך לא-קווי על קביעת יעדי איכות ועל השגתם? מבחינות רבות, סוגיות תכנון האיכות זהות לאלו של התהליכים הקוויים. בעיות מסוג מדיניות האיכות, משאבים, הדרכה ושאר סוגיות ההנהלה הכלליות, נשארות ביסודן ללא שינוי; את השינויים נמצא בדינמיקה של הפרויקטים.

תוכנית האיכות היא שדה הקרב בו מתאחד המאבק לאיכות עם הפיתוח הלא-קווי. זה המקום בו הקרבות האידיאולוגיים הפנימיים חריפים ביותר. מן הצד האחד, נמצאת התרבות המונעת על ידי תוצאות, ממנה צומחות, בדרך כלל, השיטות הלא-קוויות. בתרבות זו, כל הנראה כתקורה לא-חיונית משמש כמועמד לחיסול מיידי, וכל דבר הקשור לתכנון איכות, נראה כמטרה נוחה לחיסול. פעילויות איכות מאופיינות (בדרך כלל) כדבר איטי וקפדני, שאינו מסוגל לתרום לערך העסקי. הגישה הלא-קווית נתפשת כמי שמנחיתה מהלומה על הביורוקרטיה.

מן הצד השני, ניצבים חסידי האיכות החושדים עמוקות בשיטות המונָעות על ידי תוצאות. זאת משום שלעיתים תכופות בעבר, הן שימשו כהסוואה לגישה לא ממושמעת ולא מבוקרת לנושא הפיתוח. ההיסטוריה מראה שיישום שיטות כאלו נדון לכישלון לעיתים תכופות, אך היה גם מספר מספיק של הצלחות שדי בו כדי לא לתת

לוויכוח לדעוך. מובן שאנו רוצים להימנע מביורוקרטיה מיותרת, אך אם עד כה עלה בידינו לספק תוצרת כלשהי, הרי זה רק בשל גישתנו הקפדנית לצורך בסקירות חוזרות ובניסויים. גם אם הפעולה לוקחת זמן רב יותר, לפחות התוצאות הן בסדר, או לפחות, ככל האפשר בסדר, בהתחשב בעובדה שהמשתמשים משנים את דרישותיהם כל הזמן!

תפקיד תוכנית האיכות הוא למקד את הדיסציפלינות המגולמות במערכת איכות לתוך פרויקט ספציפי. כך, כל פעילויות האיכות החיוניות ייכללו בו, וברמה הנכונה. עם זאת, ניתן למנוע הכללת פעילויות לא-חיוניות , כברירת מחדל.

בפיתוח קווי, תוכנית האיכות היא לעיתים רק רכיב קטן בתוכנית הפרויקט, רכיב זה מכיל מספר התייחסויות לנהלי איכות בהם יש להשתמש, אך לא יותר מזה. למרות שזה כל מה שנדרש במקרים מסוימים, גישה זו מתעלמת מן הפוטנציאל האמיתי של תוכנית האיכות, המתבטא ביכולתה לשמש כתוכנית כל הפעילויות לניהול הסיכונים. בפיתוח לא-קווי חייבת תוכנית האיכות לזהות את שלבי המידול, את מנגנון ההערכה ואת המדדים שישמשו להערכת התוצאה המתקבלת מכל שלב. בתפקיד זה הופכת תוכנית האיכות לחוליה החיונית שבין הפיתוח לפעילויות ניהול הפרויקט. בתוכנית האיכות מוגדרת שזירת שני תהליכים אלה, ובה מחליטים על האיזון שבין הרצף לאיטרציה, כלומר הפעולות שחוזרות על עצמן. יצירת תהליך מבוקר ועם זאת גמיש שמספק שקיפות של ההתקדמות, אך ללא ביורוקרטיה, היא אתגר נכבד ביותר.

בתוכנית האיכות עבור פיתוח לא-קווי, הסוגיה היחידה החשובה ביותר שיש לטפל בה, היא שאלת אופי פעולת ההערכה שיש לבצע בסוף כל מחזור מידול. ההערכה חייבת לכלול קלט מכל שחקני המפתח, מצוות הפיתוח וגם מקהילת המשתמשים. עם זאת, יש לבצעה בצורה שלא תעכב את ההתקדמות. עיכובים יכולים להיווצר גם בעת ביצוע ההערכה עצמה וגם בעת יישום ההחלטות המתקבלות כפלט ממנה.

מנגנון ההערכה חייב להיות :

- כפוף ללוח זמנים מדויק ביותר, כדי שכל המשתתפים יוכלו להתפנות מראש לזמן הנדרש לביצוע ההערכה ;
- מאורגן לפי כללי התנהגות מוגדרים היטב, כדי שלא יבוזבז זמן על נושאים נהליים בזמן ביצוע ההערכה ;
- מבוסס על קריטריונים מוגדרים מראש כדי לוודא אובייקטיביות מירבית ;
- להסתיים בהחלטה, או החלטות, על השלב הבא ועל הפרויקט בכלל.

בפרקים הבאים נעסוק ביתר פירוט במנגנון ההערכה עצמו.

אימות ובדיקת תקפות

גישת מחזור החיים הקווי של סקרי התיכון (ISO 9000-3, סעיף 5.6.4) ושל הבדיקות המובנות (5.7.2), מבוססת על ההנחה שהמוצרים המוגמרים של פרויקט הפיתוח נוצרים בדרך מובנית ברמות גבוהות יותר של פירוט לפני הקידוד, וברמות גבוהות יותר של פירוט הפונקציונליות הכללית אחרי הקידוד. ברור שהנחה זו אינה תקפה

לגבי פיתוחים לא-קוויים, בהם מוחלפת הגישה המובנית על ידי גישת המידול האיטרטיבי. מה יהיה התהליך?

הדרך הטובה ביותר להשגת האימות היא באמצעות **סקירת תהליך** (process review), סקרים נפרדים של האיכות או של התיכון, דורשים זמן הקמה רב ועשויים אף לחולל פעולות הכרוכות בעבודות חוזרות, אינם עולים בקנה אחד עם אופי הפיתוח ההתפתחותי. לכן חייבים הסקרים להתבצע במסגרת הערכות המודל, ומסקנותיהם חייבות להילקח בחשבון בעת תכנון שלב המידול הבא. במקרים בהם נדרשת עבודה חוזרת, צריך יהיה לשלבה בעבודת פיתוח המודל של השלב הבא.

גם לאימות באמצעות בדיקות יש חשיבות, אלא שאי אפשר לבצעו בדרך מובנית שקשורה עם הפיתוחים הקוויים, משום שאין הוא מפיק מוצרים כלשהם פרט למודלים, ואלה נמצאים כבר בפונקציונליות המלאה, או קרוב אליה. הבדיקות הקריטיות ביותר הן בדיקות הקבלה המבוססות על היעדים העסקיים שנקבעו בתחילה, וצריכות לכלול את כל הפרמטרים הדרושים של הביצוע. בדיקות הקבלה צריכות להירשם ולקבל אישור לפני התחלת המידול.

בסוף כל שלב מידול, ניתן להריץ את בדיקות הקבלה ולציין את החריגות; אלו ישמשו מודד לגודל ולאופי הפער שבין המודל לבין הדרישות וישמשו כקלט ראשוני לתהליך ההערכה. בעת ההערכה, יכול השלב השני של המידול להתמקד בשיפור חולשות ספציפיות ובהרחבת המודל, או שניתן לעבד מחדש את המודל כדי לתת לו כיוון שונה. במקרה שההערכה מצביעה על שינוי ביעדים הכוללים, ניתן לעבד מחדש את בדיקות הקבלה כדי שישקפו את השינויים, עוד לפני בדיקת שלב המידול הבא.

האימות ובדיקת התקפות דומים בעיקרון לאלה שבפיתוח הקווי, הם שונים בביצוע בפועל. הבדל זה מורגש במיוחד בפרשנות של ההנחיות לתקן ISO 9000-3 סעיף 5.7, בו מצביעים בבהירות על גישה מובנית המבוססת על תיכון המאורגן בצורה היררכית.

ניהול התצורה

ניהול התצורה חשוב במיוחד לפיתוח התפתחותי, אך המנגנונים יכולים לדמות בתמצית לאלה של הפיתוח הקווי. השינוי היסודי ביותר הוא התהליך לבקרת שינויים, שחייב להיות משולב בהערכת המודל. בשלב ההערכה, מתקבלות החלטות על המצב הנוכחי של המודל, ושינויים כלשהם חייבים לקבל אישור לקראת שלב המידול הבא. מנגנון בקרת השינויים חייב לקיים את יכולת המעקב החל מהדרישות ההתחלתיות, דרך כל המודלים שאושרו, ועד למערכת הסופית כפי שתימסר בגמר התהליך. צפוי שיהיו מודלים שלא יאושרו ליישום. עם זאת, יש מקום לשמור חלק מהפונקציונליות שלהם או את כולה, לעיון בשלב מאוחר יותר. חיוני שכל המודלים האלה יהיו תחת בקרה, וכי ידוע יהיה השלב בו נוצרו ברצף המודלים, כדי שבמועד הנכון, אפשר יהיה לקבל החלטה על הכללתם בתהליך, באופן מלא או חלקי.

להנחיות תקן ISO 9000-3 יש ערך רב והן אינן יוצרות בעיות פרשנות חדשות כ
כאשר מיישמים אותן לפיתוח הלא-קווי. אלא שיש לזכור, שהשימוש בהנחיות
להיות משולב אפקטיבית עם התכנון הלא-קווי ועם מנגנוני הבקרה.

מה הם היתרונות והמכשולים שבגישה הלא-קווי

ככלל, הפיתוח הלא-קווי נותן למשתמשים בקרה רבה יותר על תהליך הפ
ותחושת אחריות גדולה יותר לתוצאות. לכן, רמה גבוהה של מחויבות ההנהלה
תנאי מוקדם וחיוני לפרויקטים מוצלחים. לפנינו מספר יתרונות וחסרונות.

פיתוח לא-קווי - יתרונות וחסרונות

היתרונות

1. אספקה מוקדמת יותר, אפילו של מערכת חלקית, מאפשרת הערכות מלאות
והזדמנויות רבות יותר לשיפורים.
2. הגישה מספקת תוצאות. כתוצאה מכך, במקרה של קיצוצים בתקצ
במשאבים, אין סכנה שנישאר רק עם עיצוב על הנייר שלא-מיושם.
3. ההנהלה יכולה לשנות קדימויות, או במקרה שהכל לא מצליח, היא יכולה
פרויקטים חסרי ערך, לפני שאלה יצרכו את רוב המשאבים המוקצים ש
בניית פונקציונליות שאין בה צורך.
4. סביר להניח שיהיה קל יותר לקבל אישור לפרויקטים שהם רבי ערך,
מסוכנים במקצת, תוך התבססות על כך שהסיכונים ניתנים לשליטה.
ביכולתנו להשיג שיפורים ממשיים במערכות, במקום להעסיק עצמנו ב
'בטוחותי' לכאורה.

החסרונות

1. העדר מחויבות ברורה של ההנהלה יכולה להותיר את הפרויקט ההתפתחות
תחושת כיוון ; וסביר להניח גם, שיגרום להשתתפות לא מספקת של המשתמ
2. הפיתוח ההתפתחותי מבוסס לעיתים תכופות על שימוש בכלים של
אבטיפוס. תלות-יתר בכלים אלה יכולה להביא להטיה בכיוון של פתרון
המועדף על ידי הכלים.
3. חוסר בבקרה אפקטיבית על צוות הפיתוח יכול להוליך להתרחקות
מהיעדים העסקיים, לעבר פתרונות קוסמים יותר מהבחינה הטכנית.
4. מנגנון הערכה לא אפקטיבי יכול להוביל לחזרות על שלבים ומתוך כך גב
חוזר של עבודות, מתוך רצון לעמוד ביעדים לא עקביים, או משתנים.

שיטות לא-קוויות מציעות יתרונות משמעותיים במחיר מספר סיכונים רציניים, ומהוות אתגר למנגנוני האיכות הממוסדים. כל העקרונות שנקבעו בפרקים קודמים עדיין אפקטיביים ומתאימים למטרה שקבענו. יישום השיטות בסביבה חדשה זו מחייב גישה חדשה. שני הפרקים הבאים מוסיפים לחקור נושאים אלה בהקשר של שתי טכניקות ספציפיות.

פרק 8

יצירת אבטיפוס ופיתוח יישומים מהיר

מבוא

פיתוח יישומים מהיר (Rapid Applications Development - RAD) כולל אוסף טכניקות שמטרתן העיקרית היא לאפשר הספקת יישומים למשתמשיהן במהירות גדולה יותר מזו שהתאפשרה בשיטות המובנות המסורתיות. במקרים רבים מנצלת גישה זו את השימוש בטכניקות של יצירת אבטיפוס, ובכלים המכוונים לסייע למנגנוני הספקה מהירים. עם זאת, תופעות אלו אינן תמצית הגישה. בלב הגישה מצויים פילוסופיה התפתחותית וסגנון פיתוח לא-קווי, המעניקים לגישה ביטוי מעשי. שיטת RAD עוסקת בהגבלת פרויקטים לתחום של מה שאפשרי במסגרת זמן סבירה, הספקתם במהירות תוך השתתפות מירבית פעילה של המשתמש בתהליך, התייחסות לאסטרטגיה ההתפתחותית לטווח ארוך, שתרום לה כל אחד מהפרויקטים השונים. מנקודת הראות של איכות התוכנה, RAD מהווה סטייה משמעותית ממה שהפך לקביל ומסורתי. פיתוח יישומים מהיר מהווה אתגר משמעותי לאלה שאימצו את השיטות המובנות ואת מערכות ניהול האיכות.

יצירת אבטיפוס בתהליך הפיתוח

יצירת אבטיפוס בתוכנה - ימיה כימות פיתוח התוכנה; יצירת אבטיפוס כשיטה למידול ולהערכת עיצובים עתיקה עוד יותר. לכן, הגישות של שימוש באבטיפוס בפיתוח תוכנה אינן פתרון חדשני או מקורי במיוחד לבעיות הנדסת התוכנה. ניתן לומר על שיטה זו, שמצאה את מקומה הנכון במארג הכללי של הדברים.

תיאור היסטוריית התכנות חייב לפתוח בהופעתו **כישור** (skill) חדש הקשור במחשבים. בשנים המוקדמות של פיתוח תוכנה, נחשב ייצור התוכניות לכישור המרכזי של המחשוב. כאשר התחילו להתייחס להנדסת התוכנה כאל שיטה לשיפור איכות התוכניות (ובכך גם לשיפור הפריון לטווח ארוך של התוכניתנים), הוסטה תשומת הלב מהתכנות, ששוב לא נחשב כדיסציפלינה המרכזית. השיטות המובנות שהופיעו בהדרגה תרמו להגדרה מפורשת (פורמלית) של תהליך הפיתוח, והגדילו את הסיכויים להגיע לסיום מוצלח, אך במחיר אובדן מסוים של היעילות ולוחות זמנים מוגדלים.

ככל שהוסיפה הנדסת התוכנה להתבגר, נעשו השיטות המובנות מקובלות יותר ואף הצליחו חלקית לפחות, לספק יישומים בדרך מבוקרת. עם זאת, עלות הבקרה המוגברת התבטאה בצמצום מסוים בגמישות, ופרויקטים נמשכו זמן רב יותר ועלו יותר ככל שגבר השימוש בשיטות מובנות. חשיבות מיוחדת נודעה לשתי תופעות:

1. השיטות המובנות נטו להגדיל את האמון בתהליך הפיתוח וגם עודדו את הארגונים להיכנס לפרויקטים גדולים מאלה שהיו אפשריים בעבר. כתוצאה מכך הופיעו בעיות חדשות של גודל ומורכבות.
2. לעיתים תכופות נעשה השימוש בשיטות מובנות ללא אבחנה, ללא כל ניסיון להתאים את השיטה לתרבות הארגונית, או לאופי היישומים. מצב זה הותיר את המפתחים עם תקורה מיותרת ועם עזרה מועטה, או ללא כל עזרה בבעיות הפיתוח.

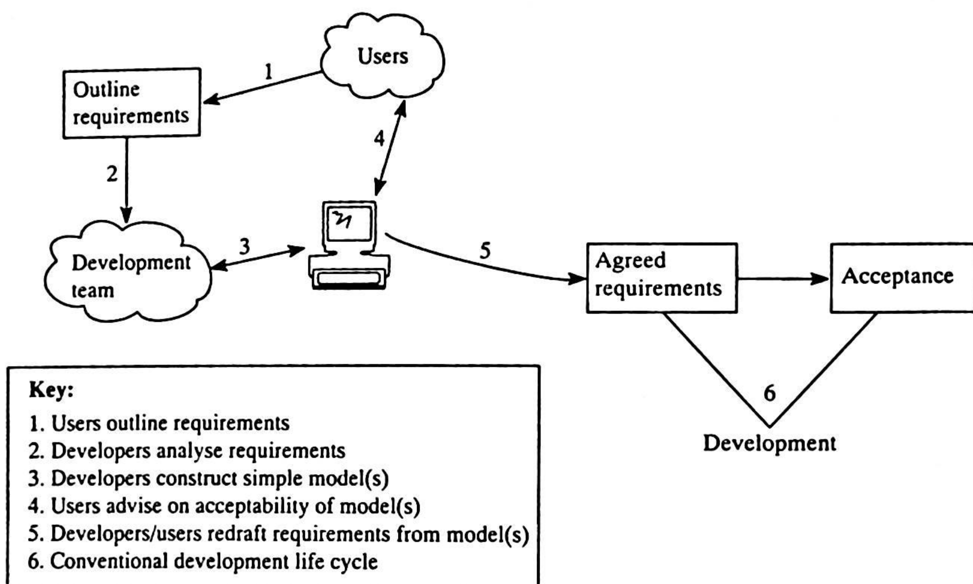
מפתחי תוכנה רבים אינם חשים בנוח עם התהליכים הפורמליים של הנדסת התוכנה. הם רואים בשימוש באבטיפוס הזדמנות להתחמק מהתהליך המובנה, וממשיכים להפיק יישומים על ידי כתיבת קוד, כמעט ללא ניסיון לתעד את המפרט התיכון שלפיו ייבנה הקוד. לשימוש באבטיפוס יצא שם רע בין אלה שיש להם מחויבות לגישה מובנית יותר. צוטטו גם דוגמאות רבות של חריגה בעלויות ובזמן, בפרויקטים של אבטיפוס שלא הצליחו להגיע לפתרון, או סטו מהמטרה לחלוטין.

באקלים של סביבת עסקים המשתנה במהירות ותובעת תגובה מהירה ממערכות המידע של הארגון, מושמעות כל הזמן טענות נגד התקורות של הפיתוח המובנה. יתר על כן, שמירה על מחויבות המשתמשים ומעקב אחר דרישות מורכבות ומשתנות על פני פרק זמן ממושך של הפיתוח, מתבררים במקרים רבים כבלתי אפשריים. מחד, אנו מוכרעים על ידי הממדים והמורכבות. ומאידך, בו-זמנית, אנו נתבעים להגיב במהירות רבה יותר לצורכי מערכת המידע.

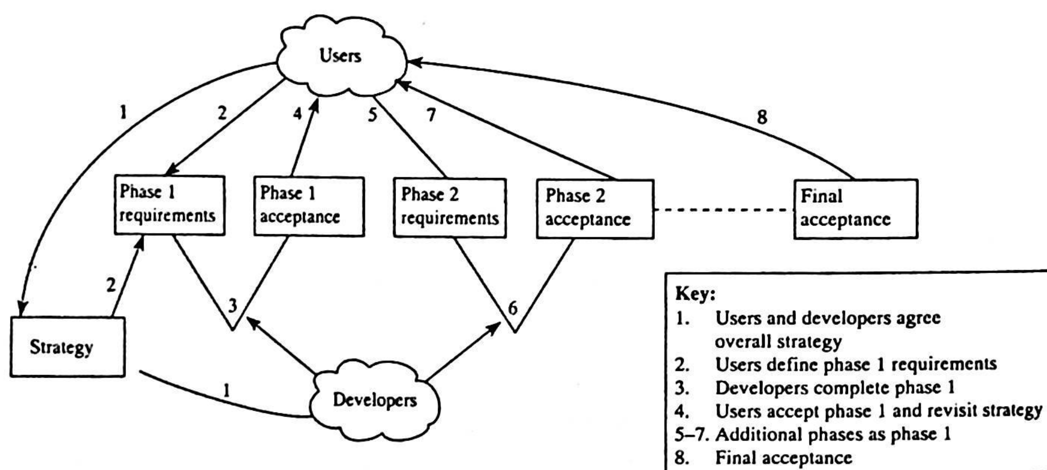
השימוש באבטיפוס במסגרת דיסציפלינרית שצמחה מתוך ניסיון בשיטות מובנות, חוזר אל הבמה בדמות **'פיתוח יישומים מהיר'** (RAD).

שני סגנונות של שימוש באבטיפוס מקובלים היום בענף:

1. **יצירת אבטיפוס של הדרישות** (תרשים 8.1), שנקראת לעיתים יצירת אבטיפוס לצורך בירורי גישוש, או לשימוש חד-פעמי. בשיטה זו משתמשים בתחילת תהליך הפיתוח כדי לברר את הדרישות עם המשתמש. לאחר מכן מניחים בצד את האבטיפוס וכותבים מפרט מלא של הדרישות. תהליך דומה של שימוש באבטיפוס משמש להגדרת ממשקי המשתמש, תוך הסתייעות בעיצובי מסך, תבניות דוחות, וניווטים במסך ובתפריטים.
2. **פיתוח התפתחותי** (תרשים 8.2) כרוך ביצירת מודלים של המערכת הנדרשת, שהם בעלי ערך הולך וגדל עבור המשתמשים. כל הספקה התפתחותית (אבולוציונית) מכוונת להיות שלמה בכל הנוגע להבנה הנוכחית של הדרישות. אלא שבהדרגה, תוך כדי רכישת הניסיון, נעשות הדרישות עצמן מפורטות יותר, והתוכנה מפותחת כדי לעמוד בניסוח החדש שלהן. גישה זו ניתן להשוות לברבור המתפתח מתוך דמות הברווזון המכוער שהיה בתחילה.



תרשים 8.1 אבטיפוס של הדרישות



תרשים 8.2 אבטיפוס התפתחותי (אבולוציוני)

פיתוח יישומים משותף

גם בגישת האבטיפוס לנושא הפיתוח עדיין אנו זקוקים להצגת הדרישות על ידי המשתמש, ודבר זה יכול להוות בעיה. הגישה אל המשתמשים הראשיים כדי לוודא שניתוח הדרישות יכסה את הסוגיות החשובות ביותר, היא כשלעצמה נושא לדאגה. גם אם נצליח להשיג מעורבות זו בשלב מוקדם, עלינו לוודא גם שנוכל להמשיך בה. הערכה מתמשכת של המערכת על ידי המשתמשים חיונית להצלחת השימוש ההתפתחותי באבטיפוס.

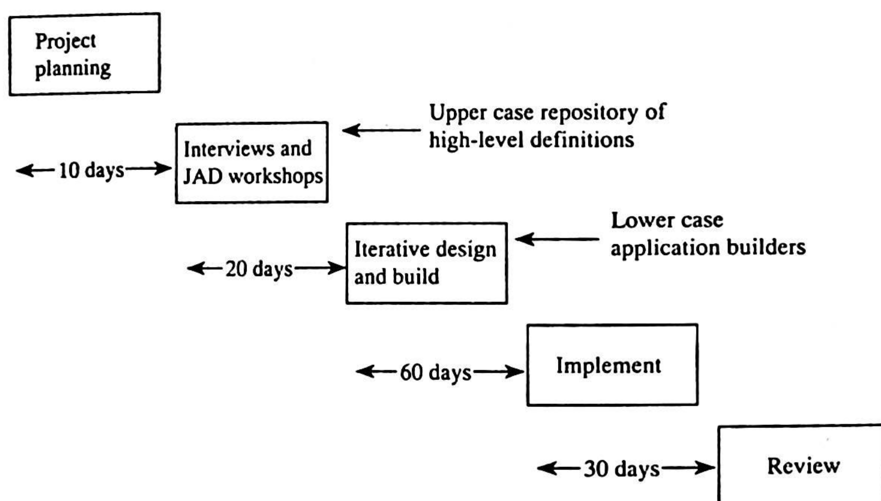
פיתוח יישומים משותף (Joint Application Development - JAD) התגלה כתשובה מתאימה לצורך במעורבות מוקדמת של משתמשי המפתח. פיתוח זה מתמקד על השגת מעורבות עמוקה של המשתמשים בתחילת הפרויקט, על ידי כך שהוא קובע פרץ קצר של אינטראקציות מובנות ביותר בין המשתמשים למפתחים. המאפיין המשמעותי ביותר של JAD הוא המנגנון להגדרת יעדים באמצעות מה שמוגדר בדרך כלל בשם **סדנאות JAD**. לנוהג זה יש מספר חלופות, אך לכולן עיקרון בסיסי אחד והוא, להעסיק את משתמשי המפתח ואת המפתחים בפעילות משותפת, שמכוונת להפקת דרישות עבור הפרויקט, ששני הצדדים יתחייבו לגביהן.

באומרנו 'משתמשי מפתח' ו'מפתחים' כוונתנו לאותם אנשים היכולים לקבל אחריות למערכת ולתהליך העסקי בו היא אמורה לתמוך. משתמשי מפתח הם מי שיכולים לקבל החלטות מחייבות בנוגע למה שהוא קביל, ומסוגלים לקבל החלטות בדרג ההנהלה במקרים שקיימת תחרות בין משתמשים שונים. מפתחי המפתח הם אלה שיש להם הידע והסמכות לקבל התחייבות בשם הארגון המְפַתֵּח.

שאלת **ההיתכנות** (feasibility) או ישימות של הדרישות מותנית במידה רבה בממדי הזמן המוקצה לפרויקט. רוב הדרישות יכולות להתבצע בפרק זמן כלשהו, אלא שבסביבת RAD עלינו להגביל את עצמנו ליישומים שניתן לספקם במסגרת לוח זמנים קצר, כזה שחופף את אופק התכנון של כל משתמשי המפתח.

במקרים רבים, המנגנון המשמש לכך הוא סדנה המונחית על ידי מתאם, שתפקידו לוודא שהסדנה תגיע לסיכום מוצלח, מבלי להיתקע בּיריבויות פוליטיות ואישיות ומבלי לפגוש במכשולים שאין לדעת איך לפותרם. כאשר קיימת נוכחות של משתמשי המפתח, יש לסדנה סמכות לקבל החלטות. הסדנה עשויה גם לשקול ויתורים הדדיים בין הדרישות של המשתמשים השונים, או בין ציפיות שהן רצויות באופן כללי אך אינן מתיישבות זו עם זו. בנוכחות המפתחים הראשיים, ניתן לקבל החלטות מחייבות בדבר ההיתכנות. בסוף הסדנה, או בסוף שלב הסדנה, במקרה של סדנאות אחדות, חייבות להתקבל דרישות מוצקות, לפחות עבור החלק הנוכחי של הפרויקט. תרשים 8.3 מציג את התהליך.

לוחות הזמנים המוצגים בתרשים אינם בלתי אופייניים לפרויקטים קטנים עד בינוניים. אלה משתמשים בטכניקת האבטיפוס כדי להשיג פתרון מערכתי קביל, שלאחר מכן מעובד לצורה המתאימה להפצה בין המשתמשים. החלופה יכולה להתבטא בהארכת השלב האיטרטיבי למשך 90 יום, בהם ייבנה אבטיפוס שיהיה ניתן למסירה בכל אחד מהשלבים.



***תרשים 8.3** פיתוח יישומים משותף - JAD*

השימוש בכלי CASE בסדנאות JAD אינו חיוני לטכניקה זו, אלא שכלים מתאימים יכולים לפשט ולזרז את שלבי הפיתוח של הפרויקט. כלים הולמים הם כלים הפועלים **כמסד מידע** (repository) עבור מידע תיכון מדרג גבוה, שמאפשר לבנות ולטפל במודלים תפיסתיים של המערכת המוצעת. מתוך מסד המידע של התיכון ניתן לבנות יישומים, לפעמים תוך שימוש בטכנולוגיית CASE, כדי לחולל תוכנה באופן אוטומטי למחצה.

התוצאה שתתקבל משלב JAD היא קבוצה מוסכמת ומתועדת של דרישות שתהווה את הבסיס לעבודת הבנייה של האבטיפוס ולמבחני הקבלה הבאים בעקבותיה.

פיתוח יישומים מהיר - RAD

בעוד ששיטת פיתוח היישומים המשותף (JAD) עוסקת במעורבות המוקדמת של המשתמשים, **פיתוח היישומים המהיר** (RAD) עוסק בעידוד המשך שיתוף פעולה של המשתמשים. את RAD אפשר לראות כהרחבה טבעית של JAD, בה משתמשים באותו תהליך בסיסי, אלא שלוחות הזמנים מוגבלים בצורה חמורה, כדי לאכוף הספקה מהירה של אבטיפוס.

העיקרון הראשי של RAD הוא היכולת לפעול במסגרת אופקי תכנון סבירים למשתמשים. יש קושי בשמירת ההתעניינות ומחויבות המשתמשים ובהשגת גישה נאותה אל משתמשי מפתח על פני מחזור מורחב של מחזור החיים של הפרויקט. קושי זה הוביל לקיצור מחזור החיים לתקופה שתימצא בתוך תחום המיקוד המיידי של המנהלים המעורבים. בדרך כלל, שישה חודשים נחשבים כגבול הזמן שלאורכו ניתן למתוח התעניינות זו.

הדרישות המוקדמות החלות על JAD חלות גם על RAD: רמת מחויבות גבוהה של המשתמשים, ונכונות לחקור את הדרישות בדרך אינטראקטיבית. גם בשיטת RAD דרוש בדרך כלל, מתאם שתפקידו לעודד דו-שיח מוקדם קריטי, ולנהל את האינטראקציה בין משתמשים למפתחים. עם זאת, בשיטת RAD מוצבות תביעות שונות לצוות המפתחים.

לוחות הזמנים הלחוצים הקשורים בפרויקטים מסוג RAD מחייבים ניהול קפדני ביותר. מנהלי פרויקטים חייבים להיות מסוגלים:

- להעריך בדייקנות את לוחות הזמנים;
- להעריך בדייקנות את הדרישות למשאבים;
- לבקר את ביצוע העבודה במסגרת הפרמטרים המשוערים, כדי לקיים את האמון והמחויבות של המשתמשים.

צוות לפיתוח יישומים מהיר

צוות הפיתוח בשיטת RAD חייב להיות בעל בסיס רחב, בעל הבנה יסודית גם בכלי הפיתוח וגם בסביבת המשתמש, ובעל כישורים בין אישיים מעבר למקובל. צוות זה צריך לכלול (ארתור, 1992):

- מומחה לכלי פיתוח, כדי לוודא את השימוש הטוב ביותר בקבוצת הכלים;
- מומחה לעסקים, כדי לוודא שהיישומים ישקפו ערכים עסקיים;
- משלב יישומים, כדי לוודא שהיישומים תואמים זה לזה במסגרת האסטרטגיה הכוללת;
- מומחה לממשק אדם/מחשב, כדי לוודא עקביות במסכים ובתיבות הדו-שיח;
- ספרן, לצורך הבקרה על המוצרים המוגמרים;
- פותר בעיות, לצורך זיהוי פתרונות המחשוב הטובים ביותר;
- מומחה כללי, כדי לוודא שלא ימציאו את הגלגל פעמים רבות מדי;
- מומחה לבדיקות, לוודא שהמוצרים המוגמרים יהיו ראויים ומוכנים לשימוש.

לא כל התפקידים האלה מחייבים מעורבות ישירה בכל פרויקט וגם אינם מחייבים אדם נפרד, אך כולם תורמים למתודולוגיה הכוללת של RAD.

המשתמשים זקוקים למינימום מובטח של מוצרים מוגמרים, כדי לעודדם לקבל על עצמם את הסיכון שביציאה לפרויקט כה דינמי. אפילו במסגרת של JAD, הצבת דרישות יציבות שאנו בטוחים בהשגתן, בפרויקט של לוח זמנים קצר, אין בה משום וודאות מלאה.

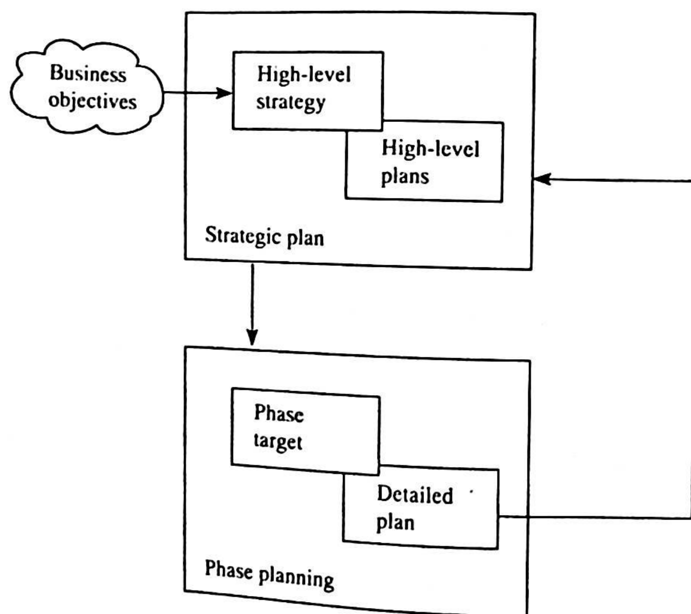
התשובה היא בקביעת פרויקטי RAD בתוך האסטרטגיה שמקיפה את המבט הרחב ביותר האפשרי, של צורכי מערכת המידע. אסטרטגיה רחבה זו מעודדת פיתוח ארכיטקטורת מידע כוללת, בה ניתן להגדיר פרויקטים יחידים של מערכות מידע. במסגרת כזאת ניתן להקצות יעדים ספציפיים לפרויקטי RAD נפרדים. תוך כדי

התקדמות הפיתוח, ניתן להתבונן בפיגורים בהספקה בתוך ההקשר הרחב יותר, ובמקרה הצורך, להפנות מקרים אלה לפרויקטים אחרים שבתוך המסגרת, כדי לצמצם את הפגיעה במשתמשים. אפילו כך, אמון המשתמשים קשור קשר הדוק להצלחה, והספקת קטלוג שלם של הדרישות המוסכמות הוא למעשה התוצאה הקבילה היחידה.

יצירת אסטרטגיה התפתחותית

דבר אינו חשוב יותר בכל אחד מענפי פיתוח התוכנה מאשר קבוצת יעדים ברורה, מכומתת ובנויה לפי סדר עדיפויות. זה נכון במיוחד כאשר שוקלים שימוש באבטיפוס כבשיטת פיתוח, משום שצוותים העוסקים ביצירת אבטיפוס זקוקים למידה מסוימת של אוטונומיה בקבלת החלטות יומיומיות, העוסקות בכיוון בו יתפתח המודל שלהם.

טום גילב (Gilb, 1988) הצביע על כך שבאבטיפוס אנו מבקשים לצעוד את הצעד הבא הטוב ביותר שאפשרי מהמקום בו אנו נמצאים בפיתוח, וכי מה שנוכל להשיג בצעד זה יוגבל על ידי זמן ומשאבים. נעצור כאשר יהיה עלינו לעצור, ונרצה להפיק את התוצאות הטובות ביותר שנוכל להשיג במסגרת הזמן והמשאבים הזמינים. כדי להגדיל את היתרונות, אנו זקוקים לחופש בקבלת החלטות תוך כדי התקדמות והתבססות על מה שהושג עד כה.



***תרשים 8.4** האסטרטגיה ההתפתחותית (אבולוציונית)*

גמישות בקבלת החלטות יפה מאוד, ובוודאי תהיה מקובלת על מנהלי פרויקטים המתקשים לתכנן ולחזות את ההתקדמות בתחילת הפרויקט. אך כך נמצא את עצמנו בתוך מעגלים ההולכים ומצטמצמים, אם לא תהיה לנו נקודת התייחסות חיצונית

כלשהי. נקודת ההתייחסות היחידה שראויה למעקב היא ההגדרה של הדבר שהמשתמשים רוצים להפיק מכל זאת במונחים תועלתיים. תרשים 8.4 מדגים כיצד הרכיבים של האסטרטגיה ההתפתחותית קשורים זה לזה.

שים לב שלא השתמשנו כאן במילה **'דרישות'**; למילה זו יש משמעויות נלוות מסוימות, ביניהן ההנחה שאלה שניסחו את הדרישות ידעו באמת מה הם רוצים. מצב זה הוא כה נדיר, עד שקשה להבין מדוע אנשים מתעקשים לנסות ולהעלות על הכתב דרישות מפורטות של המערכת, שלא לדבר על תכנון וניהול פרויקט גדול להשגת דרישות אלו. עלינו להתפשר על משהו מציאותי יותר, אם כי גם הוא אינו קל להגדרה, והכוונה היא ליעדים עסקיים.

כאשר אנו יכולים לראות פחות או יותר לאן אנו הולכים, על ידי הגדרת היתרונות העסקיים המבוקשים, אנו מוגנים לפחות מההיסחפות הגורלית המוליכה אותנו בכיוון ההפוך לגמרי. אם נוכל לתאר את היעדים העסקיים שלנו במונחים של מצייני ההצלחה הקריטיים, כפי שנדונו בפרק 3, נקבל בסיס לתכנון לטווח ארוך. טווח ארוך בהקשר הזה יכול להיות ארוך יותר מאשר הפרויקט בו אנו דנים באותה עת. אדרבא, **'מצייני ההצלחה הקריטיים'** (Critical Success Indicators - CSI) יספקו כיוון ברור להתפתחות לטווח ארוך יותר, שהפרויקט הנוכחי יתרום לה, או לספק לה קו בסיס.

השלב הבא יזהה **'גורמי ביצוע עיקריים'** (Key Performance Factors - KPF) עבור המערכת. גורמי KPF הם תכונה של המערכת, הניתנת למדידה, ותורמת לאחד או לאחדים ממצביעי ההצלחה הקריטיים (CSI). KPF מאפשרים למפתחי המערכת להבין מה מצפים מהם, כך, שהשגתם תאשר שאנו אכן מתקדמים לעבר השגת מצייני ההצלחה הקריטיים.

לדוגמה, עבור חברה לשיווק בדיוור ישיר שמקבלת 600 הזמנות ביום, מציין הצלחה יתבטא בכך שהמערכת החדשה תביא לשיפור של 10 אחוז ביכולתה הנוכחית. במצב הקיים, מועסקים ארבעה פקידי קבלת הזמנות, ניתוח התהליך מגלה ש-15 אחוז מההזמנות המתקבלות אינו מעובד מסיבה זו או אחרת. המערכת החדשה תאפשר לשישה איש לעבוד בו-זמנית. זה מרמז על שני גורמי ביצוע עיקריים (KPF):

1. המערכת צריכה להיות מסוגלת לעבד עד 660 הזמנות ליום, כשלרשותה עד שישה עובדים.
2. המערכת צריכה להיות מסוגלת לבצע קליטת הזמנות בקצב של 759 הזמנות ליום (660 ועוד 15 אחוז), כשלרשותה עד שישה עובדים.

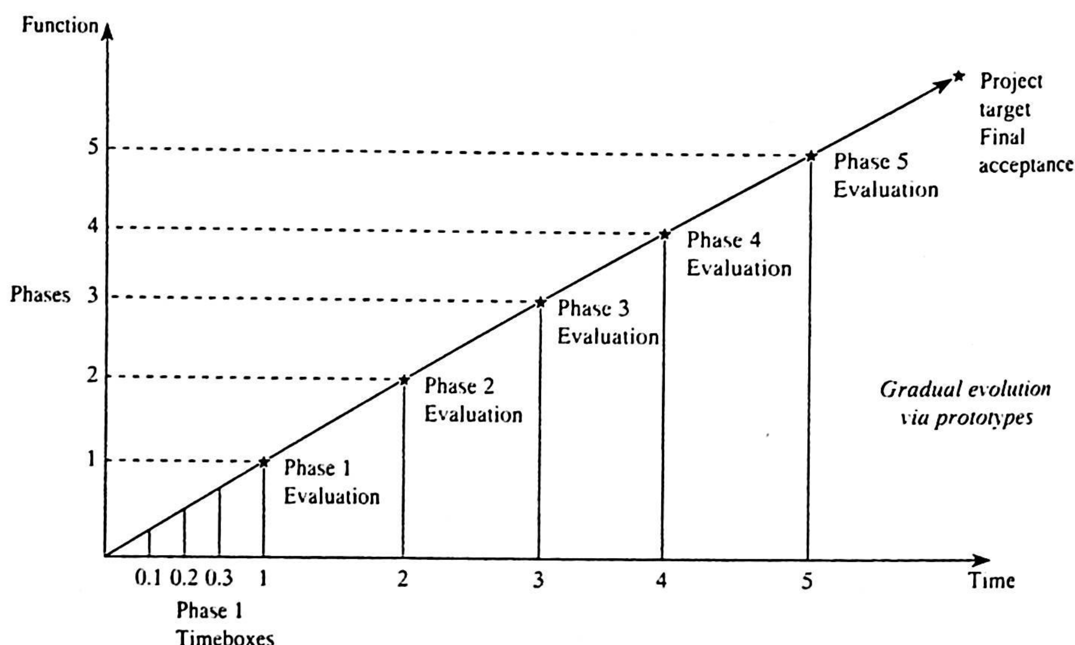
דבר אינו חדש. בפרק 3 כבר הצגנו דרך בה נוכל להשתמש בגישה זו באמצעות **טבלאות קווי פעילות** (thread tables), כדי לעקוב אחר כל התפתחות. עם זאת, מן הראוי לשוב ולציין כאן, כי גישה זו תקפה לגבי פיתוח המבוסס על אבטיפוס. למעשה, קבוצת גורמי ביצוע עיקריים הערוכה בעדיפויות, היא בסיס חיוני לדיון עם המשתמשים בעניין דרישות ספציפיות. בשלבים המוקדמים של פרויקט המובנה על אבטיפוס, בעת שאנו מנסים להרכיב תוכנית כוללת ברת-ביצוע, עלינו לדון ביעדים קצרי טווח, בקריטריונים להערכת אבטיפוס מוגמר, וב'ויתורים' שעלינו לעשות כדי להשיג את היעדים החשובים יותר.

תוכנית הפרויקט ההתחלתית תעסוק ביעדים הכלליים ותמפה קבוצה ארעית של שלבים ביצירת אבטיפוס, כל שלב והמטרה הספציפית שלו. אם הכל יתנהל כשורה, הם יספקו את גורמי הביצוע העיקריים. במקרה שהפרויקט הנוכחי הוא חלק מאסטרטגיה התפתחותית לטווח ארוך יותר, תצטרך התוכנית לזהות גם את התרומה של הפרויקט, וכיצד אפשר יהיה להעריכה מול התוכנית האסטרטגית הכוללת. בשלב זה, ניתוח יסודי מאוד של הדברים האפשריים ושל הדברים שהם ללא ספק מעבר ליכולתנו, יסייע לנו לוודא אם הציפיות מציאותיות ולהגדיל את הסיכויים שהפרויקט יספק את גורמי הביצוע העיקריים. תוכנית זו חייבת להיסקר בזהירות ולקבל את הסכמת המשתמשים לפני ביצוע עבודת תכנון או פיתוח כלשהי.

תוכנית כללית מוסכמת היא נקודת המוצא לתכנון מפורט יותר. משום שאנו יודעים שנזדקק לגמישות כדי שנוכל להתאימה תוך כדי פעולה, נגביל את עצמנו ונתכנן בפירוט את השלב הבא בלבד. התוכנית של השלב המסוים תתבסס על יעדיו שהוגדרו בתוכנית הכוללת, תפתח אסטרטגיה ליצירת אבטיפוס שתהיה מיועדת להשיג יעדים אלה, ותענה על המטרות המכומתות שיהוו את הבסיס להערכה בסוף השלב.

הערכת השלב בסופו משמשת להחלטה אם אמנם הושגו יעדי השלב. במקרה ולא, תעמודנה בפנינו שלוש אפשרויות:

- הגדלת יעדי השלב הבא, כדי להשלים כל חסר שנוצר בשלב הנוכחי;
- השלמה עם החסר, ותיכון השלב הבא כדי שיחזירנו למסלול המקורי, או קרוב אליו ככל האפשר;
- זיהוי ההשפעה שיש לחסר מהיעדים המקוריים, והחלטה על תיקון היעדים הכוללים.



תרשים 8.5 *תכנון התפתחותי (אבולוציוני)*

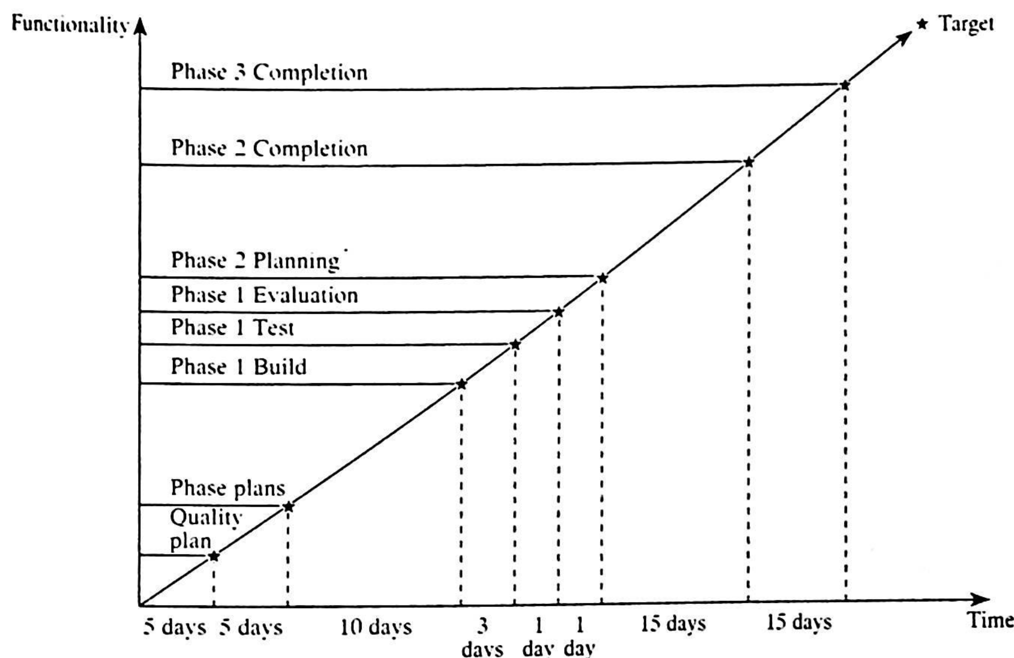
כל שלב בתהליך יכול לעצב מחדש את האסטרטגיה ההתפתחותית הכוללת, את הפרויקט הנוכחי, או את השלב הבא (תרשים 8.5). לא ניתן להטיל ספק בחשיבות שבקביעת מטרה ברורה וארוכת טווח.

תכנון פרויקט בשיטת פיתוח יישומים מהיר

כיצד ניגש לשאלת התכנון לקראת RAD? התשובה הפשוטה היא שתוכניות חייבות להתפתח יחד עם התוכנה. הדבר יקרה לא על ידי התעלמות משאלות מפתח, אלא על ידי קבלת החלטות תכנון ארעיות ומוקדמות שינחו את הפיתוח הכולל. כמו כן, עלינו לאפשר לפרטים לצוף ולעלות ככל שהפיתוח מתקדם.

תכנון ראשוני

המהלך הראשון של התכנון צריך להיות בעל טווח רחב אך לא מפורט ביותר, ועם זאת ממוקד על יעדים מוגדרים בבהירות, גם אם יש עליהם הסכמה ארעית בלבד. התוכנית הראשונה (תרשים 8.6) תבקש לזהות מה מספר השלבים ההתפתחותיים הנדרש, מה צריך כל אחד מהם לנסות להשיג, וכיצד יוערך כל אחד מהם בשעת הסיום שלו.



תרשים 8.6 תכנון ראשוני לפרויקט RAD

לתכנון האיכות יש חשיבות מיוחדת, כדי שההתקדמות בשלבים הלא-מובנים יחסית (שמשתמשים בדרך כלל באבטיפוס) תוכל להתבצע ללא ניטור יתר וללא הסיכון שבהפקת מוצרים מוגמרים בעלי פגמים חמורים. תוכנית האיכות צריכה להתייחס למרכיבים אלה:

- תקנים שיש לאמץ;
- תבנית ותוכן המוצרים המוגמרים;
- אחריות לביצוע ההערכה;
- שיטות וטכניקות בהן יש להשתמש בפיתוח;
- ניהול פעילויות יצירת האבטיפוס.

חלק גדול של ניטור האיכות יצטרך להעשות בסוף כל שלב פיתוח, במקביל להערכה העסקית של אותו שלב.

התכנון מהווה חלק בלתי נפרד של תהליך ההערכה, כדי שכל שלב יוכל להיות מתוכנן מקו הבסיס של הישגי השלב או השלבים הקודמים. יש צורך גם לסקור את התוכנית הכוללת ואת היעדים בסוף כל שלב.

בקרת תהליך פיתוח יישומים מהיר

אף אחת מהטכניקות שתוארו אינה חדשה ואינה כזאת שטרם נוסתה, ובכולן נעשה שימוש לא נכון בעבר. בבניית אבטיפוס הצליחו לעיתים ללכוד את דמיון המשתמשים עד שנמצאו גם לקוחות שביקשו לספקם להם כמוצר, ולאחר מכן גילו שהפונקציונליות החלקלקה של האבטיפוס אינה יכולה לעמוד בטיפול האכזרי של השימוש הרגיל. היו גם מקרים של הספקת אבטיפוס עם יותר מדי פוקציונליות ופחות מדי איכות. הדבר נובע מכך שתהליך יצירת האבטיפוס התמקד אך ורק על הוספת פונקציות חדשות, על חשבון הבדיקה שהיתה אמורה לוודא עוד לפני ההספקה הסופית, שהפונקציות שהתקבלו כבר, עמידות ואמינות.

בכל אחד ממקרים אלה קיים רצון כן לספק במהירות מוצר שיסב עונג ללקוח, ואמנם את הרצון הזה יש לעודד. אלא שהבעיה טמונה באמצעים בהם מטפלים במטרה זו. בפועל, לעיתים תכופות מדי, תרגיל האבטיפוס אינו תחת שליטה, והעוסקים בו מורשים לספק את מה שנראה להם לראוי. מדוע? הסיבות האפשריות רבות, כגון:

- לא הוגדרו יעדים ברורים;
- העוסקים ביצירת האבטיפוס אינם מבינים את הסביבה העסקית;
- המשתמשים אינם משתתפים בתהליך יצירת האבטיפוס;
- העוסקים ביצירת האבטיפוס אינם מדווחים למשתמשים על תוצאות ביניים;
- העוסקים ביצירת האבטיפוס נלהבים ומנסים להשיג יותר מדי בזמן שלרשותם;
- רק קומץ אנשים מבין את הכלים המשמשים ליצירת אבטיפוס.

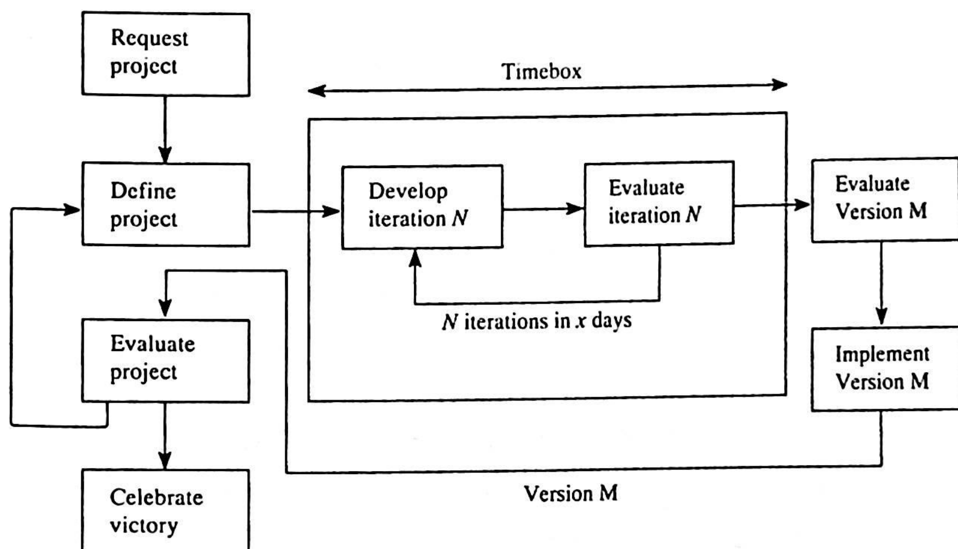
במישרין או בעקיפין, כל הגורמים האלה, עניינם בבקרת התהליך ליצירת האבטיפוס. בקרה זו היא הסוגיה החמורה ביותר בה יש לטפל לפני שאפשר יהיה לנצל את שיטות השימוש באבטיפוס.

יצירת תוכנית מדורגת, מלווה ביעדים חד-משמעיים עבור כל שלב, די בה כדי לספק את המסגרת לבקרה אפקטיבית. במשך כל שלב, צריך לבקר את הכנת האבטיפוס על בסיס יומיומי שוטף, כדי לנסות להשיג את היעדים של כל אחד מהשלבים.

זוהי בעיה סבוכה. כל ניסיון לבקרת תהליך שנועד להיות גמיש מחד, ומאידך נותן חופש לחקור מספר רב ככל האפשר של אפשרויות פעולה לעוסקים בתהליך, מסתכן בהחנקת הגמישות. לעומת זאת, כאשר מפקידים את יצירת האבטיפוס בשלמותה בידי הנחשבים ל'מומחים' לזה, רבים הסיכויים שהפעולה תסטה מהיעדים הרצויים, ותגרום לבזבוז זמן ומשאבים בחיפוש עקר אחר הפתרון הלא-נכון.

כפי שקורה לעיתים כה תכופות, התשובה יכולה להיות במציאת פשרה. הבקרה תתאפשר כאשר אופק הזמן קצר ביותר. כך ניתן להניח לעוסקים בנושא האבטיפוס לערוך בדיקות ללא פיקוח או השגחה, במסגרת הפרמטרים שנקבעו על ידי התוכנית המיידית. טכניקה זו ידועה כשימוש ב'**תיבות זמן**' (timeboxing).

גישת תיבות זמן לוקחת שלב מתוך התוכנית ומפצלת אותו לקטעי עבודה קטנים יותר, לתיבות זמן. בכל תיבה צריך להיות מידה מספקת של אוטונומיה וניתן לזהות בה מוצר מוגמר כלשהו. לעוסקים ביצירת אבטיפוס ניתנת תקופת זמן קבועה, תיבת זמן, בה עליהם להפיק את המוצר הרצוי. עם זאת, הם נדרשים להפסיק לעבוד בסוף תיבת הזמן מבלי להתחשב בתוצאות מאמציהם. בסיום כל תיבת זמן מתבצעת הערכה של ההישגים ומתקבלת החלטה בדבר תיבת הזמן הבאה.



תרשים 8.7 גישת תיבת הזמן

השימוש בתיבות זמן (תרשים 8.7) הוא בתמציתו רמת תכנון נוספת. כעת יש לנו החל רצף של תוכניות ברמות גדלות והולכות של פירוט, מהיעדים ההתפתחותיים שלטווח ארוך, ועד לרמה המאפשרת לעוסקים באבטיפוס לפעול בתוך מסגרת מוגדרת היטב, כדי להשיג מוצר מוגמר בתקופת זמן נתונה. למרות שבתיאור זה יש הצגה פשוטה של הנושא, והוא משמיט מספר רב של קשיים מעשיים, יש בו עם זאת, גילום של העיקרון הבסיסי הבא לוודא שכל עבודה שמאושרת בתוך הפרויקט, תתרום במופגן ליעדים הכוללים. מבחינה זו לפחות, השימוש באבטיפוס אינו שונה מהגישות המובנות המסורתיות יותר.

ניהול איכות

בסגנון הגישה החופשית היחסית הזו, איזה מקום נותר **לבקרת איכות** (quality control), או **לאבטחת איכות** (quality assurance)? הגישות המסורתיות של סקרים (reviews) ושל בדיקות מובנות אינם נראים מתאימים לסגנון זה. לא סביר להניח שתיכון אבטיפוס יפיק אוסף של תוכניות נפרדות שאפשר לנסותן כל אחת בפני עצמה ולשלבן יחד לאחר מכן. גם הכללתן בתהליך של סקרי תיכון, כדי שישמשו כאבני דרך לצורך אישור המוצרים, אינה נראית מתאימה.

יש להביא בחשבון שגישת האבטיפוס מלווה בהכרח בסיכונים. בסביבת פיתוח אשר נועדה במפורש להימנע ממגבלות הגישה המובנית, אי טיפול בבעיות האיכות יהיה בלתי אחראי וגם לא-חכם. דרושה גישה חדשנית, שתהלום את תרבות וסגנון השימוש באבטיפוס, ועם זאת, תספק ניהול סיכונים אפקטיבי.

תוכנית איכות עבור פרויקט שיטת פיתוח יישומים מהיר (RAD), מחייבת דגשים מיוחדים ואמורה לכסות עשר נקודות כמפורט להלן:

עשר נקודות עבור תוכנית איכות ל-RAD

1. תחומי אחריות עבור:
   - הגדרת מדדים לקבלה;
   - הגדרת תיבות זמן;
   - אישור מוצרים מוגמרים בסוף שלבים;
   - בניית אבטיפוס;
   - בדיקות אבטיפוס;
   - ניהול הספרייה;
   - ניהול הכלים.
2. קריטריונים לקבלה.
3. תוכנית בדיקות קבלה.
4. תוכניות בדיקות שלב.

5. הערכת השלב ואסטרטגיית הסקירה.
6. מנגנון הערכה.
7. ניהול התצורה, ובמיוחד בקרת שינויים.
8. דיווח בעיות.
9. כלים שיש להשתמש בהם.
10. סביבת הפיתוח.

אחד האלמנטים החשובים של גישת האיכות הוא יצירת תקנים מקיפים עבור עבודה באבטיפוס. העוסקים בעבודה זו צריכים לדעת כיצד צריכים להיראות מוצרים מוגמרים שלביים ואיזה תיעוד יש להפיק בהם. הם גם יצטרכו לדעת לפי איזה כללי תיכון עליהם לפעול ומה צריכה להיות תבנית הקוד.

סוגיית איכות חשובה נוספת היא שאלת **כמות הבדיקות** שצריכה להתבצע על כל מוצר. יש להגדיר אסטרטגיית בדיקות שתכלול את: מפתחי האבטיפוס שבודקים את המוצרים שהם מייצרים, המשתמשים שבודקים את המוצרים בשלב מתאים, ואת בדיקות האיכות שיתבצעו בשלבי מפתח של הפרויקט. כל פעילות הבדיקות צריכה להתנהל במשך שלבי ההערכה, כאשר משהים את הפעילות באבטיפוס.

לסקרים שמור עדיין תפקיד בפיתוח של אבטיפוס. בדיקות איכות הבוחנות את ההיצמדות לתקנים מסייעות לוודא עקביות ושמירה על גישה ממושמעת. סקרי איכות חייבים להוות חלק מההערכה המתבצעת בסיום כל שלב, כדי להקנות אמון ביציבות התיכון, המוצרים ובכיוון העסקי הכללי.

התחום הקריטי מכולם בפיתוח ההתפתחותי הוא תחום **בקרת השינויים**. **ניהול התצורה** אינו רק דיסציפלינה חיונית, יש הכרח ליישמו בצורה שתאפשר לבצע שינויים במהירות ובקלות, ועם זאת גם תחת בקרה רשמית. פיתוח שיטות אלו ויישומן האפקטיבי במסגרת כל פרויקט, חייבים למצוא את ביטוים המלא בתוכנית האיכות, כדי לוודא שכל המעורב בפרויקט יבין באופן ברור את המנגנונים בהם יש להשתמש.

האם פיתוח יישומים מהיר (RAD) הוא עוד אופנה חולפת (FAD)

רעיונות לשיטות פיתוח חדשות באים וחולפים. רק מעטים שורדים לזמן רב. השימוש באבטיפוס נמצא כבר זמן רב בשטח, ולא נראה שהוא עומד להיעלם, ולו רק משום שהוא מוצא חן בעיני אלה הרוצים להאיץ את הספקת התוכנה. עם זאת, לאבטיפוס יש היסטוריה רבת תהפוכות והמאבק לזכייה בהכרה עדיין נמשך.

פיתוח יישומים מהיר ופיתוח יישומים משותף (RAD ו-JAD, בהתאמה) מייצגים תדמית קבילה של השימוש באבטיפוס. כלומר, אלו הן גישות מובנות להספקה מהירה, שמכילות את הניהול והבקרה הדרושים לאבטחת אמון מסוים בתוצאה.

מהן סוגיות המפתח שהועלו על ידי RAD ו-JAD?

עשר סוגיות איכות לפיתוח יישומים מהיר

1. **דע לאן אתה הולך**

פעל תמיד בתוך אסטרטגיה ברורה וכוללת בעלת יעדים מוגדרים.

2. **דאג לכך שהפרויקטים יהיו קצרים**

הישאר בתחום אופק התכנון של המשתמשים שלך, פחות משישה חודשים.

3. **בחר בקפדנות את הצוות ליצירת אבטיפוס**

דרושים להם כישורים טובים ביחסים בין-אישיים, כישורים טכניים כ
והבנה טובה בנושאים עסקיים. אידיאלית, הם אמורים להיות מסוגלים לה
פני המים, אבל יהיה עליך להתפשר בתחום הזה.

4. **עקוב מקרוב אחר פעולת יצירת האבטיפוס**

קבע יעדים ברורים עבור כל שלב, דאג לכך שהשלבים יהיו קצרים, והש
שלב בזמן.

5. **תכנן תוך כדי ביצוע**

דאג שהתוכנית תהיה בבדיקה כל הזמן, והיה מוכן לתכן מחדש את כל הפר

6. **אבחן ופתור מראש את כל בעיות האיכות**

וודא שיוחלט מראש מה לעשות בכל הבעיות הקשות עוד לפני תחילת ה
הקצה תחומי אחריות לכל ההחלטות האמורות להתקבל במשך הפרויקט.

7. **הנח להנהלת הפרויקט לנהל**

מנהלי פרויקטים זקוקים לסמכויות כדי שיוכלו לשלוט בפעילויות. אם נ
שהם אינם ברמה הדרושה, הדרך אותם, או הבא מנהלים חדשים.

8. **דאג לכך שבדיקות הקבלה יהיו ברשותך לפני שתתחיל בעבודה**

אתה חייב לדעת אם כל אבטיפוס בפרויקט מתקדם בכיוון הנכון. תוכל לדע
על ידי השוואה למודל המוצר הסופי שברשותך, כלומר לבדיקות הקבלה.

9. **בחר בכלים הנכונים**

אם אתה משתמש בכלים, עליהם להיות הכלים הנכונים ולהימצא תחת
מתאימה. צוות האבטיפוס חייב לדעת כיצד להשתמש בהם ביעילות.

10. **וודא שההנהלה שלך נמצאת 'בראש אחד' אתך**

אם אין לך את המחויבות של ההנהלה, אתה מסתכן בהפסקת הפרויקט
שהדברים יתנהלו שלא לפי התוכנית, וזה דבר צפוי בהחלט.

השימוש באבטיפוס למן ההתחלה, אינו הדרך היחידה לבנייה מהירה של יי
שימוש חוזר במה שבנינו לפני כן, מציע נתיב פורה עוד יותר ובטוח יחסית,
לכיוון פיתוח יישומים מהיר (RAD). זהו התחום של פיתוח מכוון-עצמים.

פרק 9

פיתוח מוכוון-עצמים

מבוא

מה כל הרעש הזה בנוגע לשיטות מוכוונות-עצמים (OOM - Object-Oriented Methods)? האם השיטות המובנות אינן פועלות יותר, או אולי גילינו פתאום את סוד הגישה אל כל מערכות התוכנה שהובטחו לנו תמיד, אך מעולם לא נמסרו לנו? אדם מציאותי יודע שאין אחיזה לטענות ממין זה. יש סיבות רבות לכך ששיטות מוכוונות-עצמים ראויות לתשומת לב מסוימת, אך יש סיבה אחת מכרעת שבגינה עלינו לשקול מעבר לשיטה מוכוונת-עצמים. שיטה זו מביאה את המתרחש בנבכי המחשב קצת קרוב יותר אל המתרחש בעולם המציאות, כפי שאנו מבינים אותו. כל דבר שגורם למחשבים ולתוכנה שיהיו מובָנים יותר למשתמשיהם ראוי לעיון רציני, גם אם הוא גורר בעקבותיו מספר בעיות.

מהו עצם? התשובה מותנית במי שהנך

אם אתה מפתח תוכנה, רבים הסיכויים שהעצם (אובייקט) ייראה לך כדרך הסתכלות בתוכניות המאגדות את הנתונים והתהליכים הקשורים להם במבנה אחד. הרעיונות הבסיסיים הם הרחבה טבעית של גישות מוקדמות יותר שהתמקדו במבני נתונים, או במבני תהליכים. הדבר היה בגדר חזון עד שפותחו שפות **'תכנות מוכוון-עצמים'** (Object-Oriented Programming - OOP) שאיפשרו לבנות תוכניות ומערכות תוכנה הפועלות על פי מודל תכנות חדש זה ועל פי נהלי **'עיצוב מוכוון-עצמים'** (Object-Oriented Design - OOD).

לעומת זאת, אם אתה משתמש, סביר שתרגיש יותר בנוח בתפיסה שהעצמים הם דרך למידול העולם המציאותי. כלומר, מודל הלוכד את המאפיינים הבלתי-משתנים של הדברים בהם אנו מעוניינים ואת יחסי הגומלין הדינמיים שביניהם.

האם יש מְתָאָם בין שתי השקפות אלו? אין ספק בכך. השקפה מוכוונת-עצמים על העולם מספקת דרך לניתוח מצבים והכרה בהזדמנויות לניצול עיצוב מוכוון-עצמים (OOD) ותכנות מוכוון-עצמים (OOP). **'ניתוח מוכוון-עצמים'** (Object-Oriented Analysis - OOA) משלים באופן טבעי את הטכניקות האחרות.

טכניקות מוכוונות-העצמים הולכות ונעשות שכיחות בענף התכנות, אך טרם הגענו אל השלב בו תוכלנה טכניקות אלו להיחשב כחלופה מלאה לגישות המבניות בענף זה. שיטות וטכניקות שנועדו לאפשר את השימוש והתמיכה בגישה המובנית לנושא פיתוח התוכנה, היו זמינות במשך עשרות שנים, אך גישה זו טרם הצליחה לחלחל ולהגיע אל כל תעשיית התוכנה. מעטים הסיכויים שבטווח הקצר שיטות מוכוונות-העצמים תצלחנה להתבסס היטב, למרות היותן 'הזינוק הגדול קדימה' כטענת חסידיהן.

לרתיעה טבעית זו נוסף הקרע הקיים עדיין בין כמה מתומכי גישת תכנות מוכוון-עצמים (OOP), שרואים בה בעיקר, או רק, שיטה להשגת מאפיינים מסוימים בתוכניותיהם; לבין המספר הגדל והולך של משתמשים עסקיים שרואים את הניתוח מוכוון-העצמים (OOA) ועיצוב מוכוון-העצמים (OOD) כדרך למידול עולם המציאותי, אך יודעים מעט מאוד, או לא כלום, על תכנות מוכוון-עצמים (OOP), או על השיטות הנלוות לגישה זו. נראה שיש הצדקה להנחה ששיטות מוכוונות-העצמים יכולות לספק מעבר 'ללא תפרים' מהעולם המציאותי אל יישום תוכנה בעלת מאפייני איכות חיוביים, אלא שעד עתה היתה רק התקדמות מעטה בהמחשת פוטנציאל זה.

מספר מונחים שכיחים בשימוש בעולם מוכוון-העצמים

- **עצם** (object). הפשטה של ישות עסקית בעולם המציאותי, או של יישות טכנית.
- **עצם עסקי** (business object). הפשטה של ישות עסקית, כגון לקוח או הזמנה.
- **עצם טכני** (technical object). הפשטה של ישות טכנית מהרמה הנמוכה, כגון רשימה או שולחן.
- **מחלקה** (class). אוסף של עצמים בעלי תכונות משותפות.
- **שיטה** (method). פעולה המתבצעת על ידי עצם או על העצם.
- **הודעה/מסר** (message). פרוטוקול המורכב משיטה והפנייה לעצם. אינטראקציה בין עצמים מתקיימת על ידי העברת הודעות שדורשות מעצמים אחרים להפעיל את שיטותיהם.
- **הסתרת מידע** (information hiding). הפרדת הייצוג של עצם או של מחלקה, מפרטים הממשק לציבור.
- **הפשטה** (abstraction). סילוק הפרטים הלא-חיוניים מהמודלים, כגון סילוק המפרטים, כדי להגביר את היכולת להבינם.
- **הצמדה דינמית** (dynamic binding). קישור בין מזהה לעצם בזמן הרצת המערכת.
- **הורשה** (inheritance). מערכת קשרים בין שתי מחלקות עצמים, שבה אחת המחלקות לוקחת את כל התכונות הרלבנטיות של האחרת.
- **בסיס נתונים מוכוון-עצמים** (OODB). בסיס נתונים המבוסס על מודל נתונים בו מיוצגת כל ישות של עולם המציאות על ידי עצם של מודל נתונים יחיד.
- **ניתוח מוכוון-עצמים** (OOA). שיטת ניתוח המתמקדת במידול ישויות של עולם המציאות כעצמים בעלי מבנה ומספקים שירותים.

- **עיצוב מוכוון-עצמים** (OOD). שיטת עיצוב למידול מערכות תוכנה כאוסף של עצמים משתפי פעולה.
- **תכנות מוכוון-עצמים** (OOP). שיטת פיתוח המבוססת על עצמים שמופעלים על ידי כל חלק של המערכת, ולא על פונקציות.

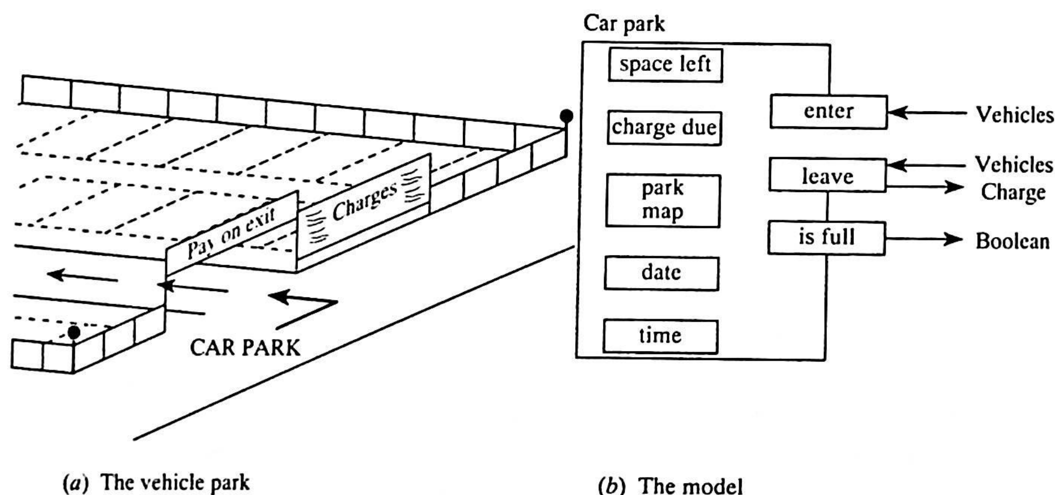
מטרת הטיפול בשיטות האיכות בספר זה היא לספק בסיס לדיונים בסוגיות האיכות. אין הוא מכוון לשמש כחומר הדרכה, אם כי הוא דן במספר היבטים של הגישה בפירוט יחסי, כדי שיוכל להצביע על יתרונות וחסרונות הגישה מנקודת ראות של איכות התוכנה. חשוב מזה, עלינו לדון בסוגיות המעשיות בהן עלינו לעסוק במהלך פרויקט המבוסס על גישת פיתוח מוכוונת-עצמים.

טכניקות מוכוונות-העצמים העיקריות מוסברות בתחילה, תוך התייחסות לרעיון המקובל של **מודל עצם** (object model).

מהו מודל עצם

למודל העצם (object model) אמורות להיות שלוש תכונות ברורות:

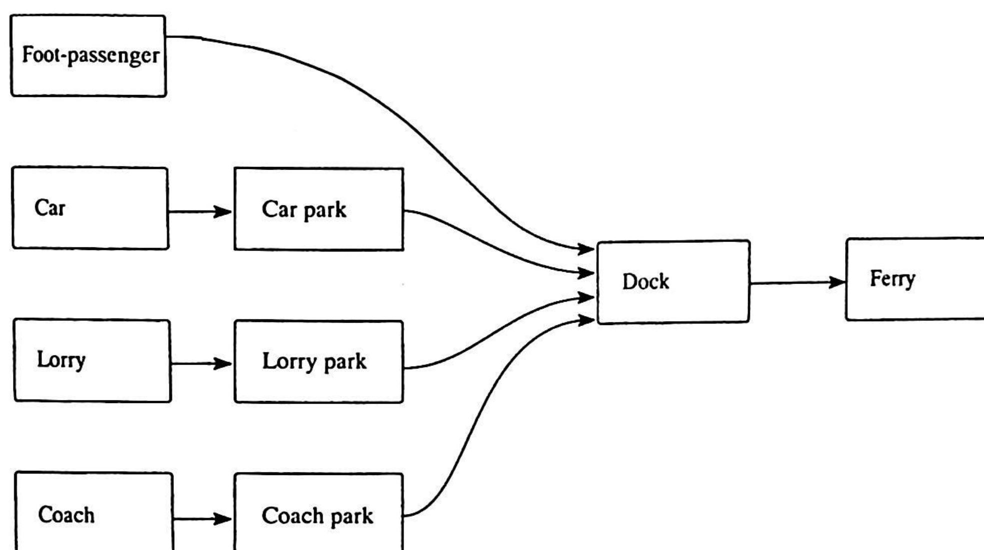
1. עליו להוות הפשטה טובה של העצם שהוא מייצג. כלומר, עליו להיות פשוט, אך בלי להשמיט פרט חיוני כלשהו.
2. עליו להיות ניתן ליישום בסביבת החומרה והתוכנה שתשמשנה לבניית המערכת שהוא מייצג חלק ממנה.
3. עליו לספק תמונה ברורה וקריאה של העצם המיועד ושל הדרך בה יוכלו משתמשי העצם לתפעלו.



תרשים 9.1 מודל עצם של חניון (a) החניון ; (b) המודל

את הרעיון של מודל עצם קל יותר לתאר על ידי התייחסות לתמונה כדוגמת זו שבתרשים 9.1, שמייצגת הפשטה פשוטה של מגרש חנייה, שבו כלי רכב מכל סוג שהוא, יכולים לחנות בכל עת.

מודל העצם שבתרשים 9.1 מייצג עצם יחיד. למידול בעיות אמת, אנו זקוקים למודלים כאלה המייצגים אוספי עצמים ואינטראקציות שביניהם, כדוגמת זה שמוצג בתרשים 9.2, בו הוחלף חניון כללי בחניונים נפרדים למכוניות, לאוטובוסים ולמשאיות. תרשים 9.2 מראה את יחסי הגומלין הדרושים בין העצמים כדי ליצור מודל של ההתנהגות בה אנו מעוניינים. התרשים אינו מראה את פרטי הממשק של כל עצם שעשוי להוות את השלב הבא של פיתוח המודל.



***תרשים 9.2* מודל עצם של מסוף מעבורת**

ניתוח מוכוון-עצמים

מטרת **ניתוח מוכוון-עצמים** (OOA) היא ליצור מודל עצם שממנו ניתן להתחיל בתיכון. זו אינה משימה פשוטה ומיידית. יש לה חשיבות יסודית, משום שהמבנה העתידי של המערכת (ארכיטקטורת המערכת) יותנה במודל המציאות שממנה הוא נבנה. מנתחי המערכת צריכים להבין את הבעיות שעומדות בפניהם ולהיות מסוגלים לנתחן במונחים של העצמים החשובים ביותר, מאפייני המפתח שלהם, ושל יחסי הגומלין והפעולות שבין עצמים אלה.

במרוצת הזמן נוצר מגוון של רישומים ושיטות לטיפול בשלב קשה זה שבתהליך.

תיכון מוכוון-עצמים

תיכון מוכוון-עצמים (OOD) מספק את החוליה החיונית בין המבט מוכוון-העצמים על העולם, לבין אוסף של תוכניות מוכוונות-עצמים. עיצוב זה קובע את ארכיטקטורת המערכת, שהיא המבנה היסודי של המערכת והמסגרת לכל יחסי הגומלין בין העצמים.

היכולת לבנות מודלים משופרים בעלי אופי אינטראקטיבי של רבות ממערכות עולם המציאות, תרמה באופן משמעותי להתגברות העניין במערכות מוכוונות-עצמים. בעיות לא היררכיות תצמחנה באופן טבעי מודלים לא-היררכיים של המציאות, שמהם תעוצב בסופו של דבר מערכת תוכנה. אם כן, מערכת מוכוונת-עצמים תהיה בנויה **אחרת** מאשר מערכת המאורגנת בצורה היררכית.

כאשר מודל העצם משקף את המאפיינים החשובים של עולם המציאות, וארכיטקטורת המערכת מאפשרת יישום מוכוון-עצמים של המודל, יש סיכויים רבים יותר שמערכת התוכנה שתתקבל אכן תהיה תואמת לבעיה.

ארכיטקטורת מערכת מתאימה מאפשרת להשיג פתרון המתאים היטב לבעיה, שמיש ומובן למשתמשים, ומגלם בתוכו מבני תוכניות טובים ועקרונות תכנות יציבים. לבוא ולומר שמערכות מוכוונות-עצמים משיגות כבר עתה מטרה זו במלואה, יהיה משום תיאור מוגזם של המצב, אך אין ספק שהדבר יושג בעתיד.

תכנות מוכוון-עצמים

תרשים 9.1 מציג אחדים מרעיונות המפתח. המשבצת הגדולה מייצגת את גבולות העצם. כל דבר שהוא במסגרת גבולות אלה נחשב כשייך רק לתחומו של העוסק בפיתות. המשבצות הקטנות יותר שמגשרות על פני הגבולות מייצגות את הממשק של העצם עם סביבתו. כלומר, אלה הם המנגנונים שבאמצעותם מקיים העצם את הקשר עם סביבתו. המשבצות שעל הממשק מייצגות 'שיטות' ציבוריות (public 'methods'), שמהוות את הדרכים הסטנדרטיות בהן יכול המשתמש בעצם לפנות אל המידע וההתנהגות שיוצרו כמודל על ידי העצם. אין דרך אחרת לקבלת גישה אל העצם. בין השיטות הציבוריות ייכללו האמצעים לכניסה וליציאה מהחניון, ולקביעה אם החניון מלא.

בתוך גבולות העצם נמצאות פיסות מידע, שמרכיבות יחד מודל סטטי של המאפיינים המבניים של החניון. החשוב שבהם הוא שלחניון יש מספר מירבי של מקומות חנייה לכלי רכב. יכולות להיות לנו גם כמה 'שיטות' פנימיות (פרטיות 'private 'methods) לתפעול מידע זה. השיטות הפנימיות הן פרטיות למפתח, ואינן נגישות למשתמשים אחרים בעצם; אפשר להשתמש בשיטות אלו כדי לעדכן את המודל, או כדי לספק חלק מהפונקציונליות שביסוד הממשק הפשוט של המשתמש.

אחד מיתרונות המפתח של **תכנות מוכוון-עצמים** (OOP) הוא היכולת הזו לבנות תוכניות המשקפות את מודל העצם בצורה שמאפשרת למידע המבני להיות **מוסתר** ממשתמשי העצם. כך, שאפשר לטפל במידע זה רק באמצעות השיטות הציבוריות

שסופקו. הממשק חייב להיות מתועד באופן מלא, כדי שאפשר יהיה לנצל את יכולות העצם במלואן, וכדי לוודא שמשתמשים פוטנציאליים יבינו בדיוק מהי דרך הגישה אל השיטות של העצם. זה חל על מודלים בכל הרמות, ובמיוחד ברמת התכנות, בה מיושם הממשק בתוכנה כדי שיגיב רק להודעות שמכילות את הפרוטוקול הנכון.

הבעיה של גישות התכנות המסורתיות היתה שניתן היה לבצע בהן שינויים, שבדרך כלל גם התבצעו, על ידי שינוי ישיר של המודל. במרוצת הזמן הפך המודל למורכב וממוקד יותר בסוגיות מסוימות שגרמו לשינויים הקודמים, עד שנעשה קשה לבצע את השינוי הרצוי הבא שנדרש במודל. גרוע מכך, יישום תוכנת המודל נעשה מורכב יותר ויותר, ותמיד היתה קיימת הסכנה שהשינוי ישפיע על שינויים קודמים ויביא לידי תוצאות לא צפויות.

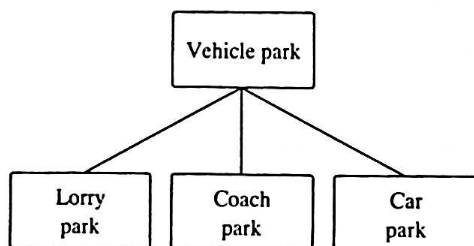
הגישה מוכוונת-העצמים, עם כל כוח המשיכה שבה, אינה ניתנת ליישום מלא בשפות תכנות מסורתיות, שאינן בנויות סביב תפיסת העצמים. שפות מובנות מסוימות התפתחו עד לנקודה בה הן מאפשרות הגדרה של מבנים דמויי עצמים, אך הכימוס (encapsulation) או 'אריזת המשאבים', היא רצונית בלבד. אין דרך לאכיפת גבולות העצם שתאלץ את המשתמשים לבצע את הגישה אליו רק דרך הממשק. שפות תכנות מוכוונות-עצמים עוצבו במיוחד, כדי להשיג תפיסת מבנה חדשה זו.

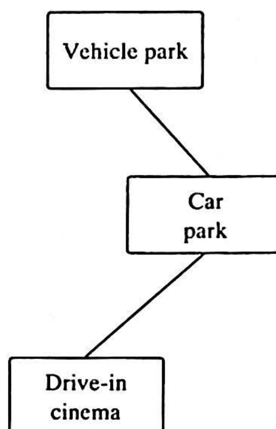
מובן שיידרשו שינויים במודל כדי להרחיבו ולשפרו, אך אסור להניח לשינויים אלה לפגוע בפונקציונליות שהושגה כבר. היכולת לאפשר לשינויים להשפיע על פיתוחים בעתיד, אך לא על ההתנהגות הקיימת, מחייבת גישה חדשה שתאפשר לשינויים להשפיע על ההתנהגות העתידית, ועם זאת לא תשתקף אחורה בזמן. זוהי התפיסה של **ההורשה** (inheritance).

ההורשה היא המנגנון שמאפשר ל'העביר הלאה' תכונות, או מאפייני עצם אחד אל עצם אחר. בעצם היורש אפשר לשנות, או לדרוס את המאפיינים שהתקבלו בהורשה. כך ניתנת היכולת לבנות צורות שונות של עצם נתון, או להרחיב פונקציונליות של עצם באמצעות תוספות. תרשים 9.3(a) מדגים כיצד ניתן היה לשנות את העצם של החניון הכללי, לחניון מכוניות, לחניון משאיות ולחניון אוטובוסים; תרשים 9.3(b) מדגים כיצד ניתן להרחיב את החניון לקולנוע 'דרייב-אין'. עם ההזדמנויות שמספק מנגנון ההורשה ליצירת שיטות חלופיות, צומחת הזדמנות חדשה. בחניון שמשמש גם למכוניות וגם למשאיות, עלינו ליצור הבחנה בין כלי הרכב, כדי שנוכל ליישם את השיטות הנכונות. לדוגמה, כאשר מגיעה מכונית, עלינו לשלוט בכניסתה באמצעות שיטת 'כניסת מכוניות' ולא בשיטת 'כניסת משאיות'. ההבחנה בין השתיים מחייבת תוספת לוגיקה, כדי שתוכל להיקרא השיטה הנכונה, וכל אלה יצטרכו להיות מקודדים לתוך התוכנית לפני ההידור שלה. כאן באה לעזרתנו גישת **ריבוי-צורות** (polymorphism).

למושג ריבוי-צורות יש חשיבות בפיתוח מוכוון-העצמים. הוא מאפשר לנו ללכוד את **ההיבטים הגנריים** (generic aspects, כלליים) של תהליך ברמה אחת, ולהאציל כלפי מטה פרטי יישומים ספציפיים של התהליך הגנרי, שלעיתים תכופות מערפלים את ההיבטים הגנריים וגורמים לבלבול. לדוגמה, תהליך הדפסת פלט המשותף למספר רב

של יישומי תכנות. הפרטים המתייחסים להדפסת מסמך טקסט, דמות גרפית או פלט מובנה, כגון גיליון אלקטרוני, שונים מאוד זה מזה. אך בכל מקרה, הכוונה הכללית זהה. אם נוכל לאפשר לתוכניתן היישום להצביע רק על כך שנדרש תהליך הדפסה, תוכנית היישום תהיה פשוטה וברורה יותר מאשר אילו היינו צריכים לכלול בה את כל הפרטים של תהליכי ההדפסה השונים. בפועל, הפרטים קשורים אל סוג העצם שדורש את התהליך, וכך אנו יכולים להאציל את פרטי התהליכים השונים להגדרה של סוגי העצמים.





תרשים 9.3 *הורשה: (a) יצירת חניון לפי סוגי המכוניות;*
(b) יצירת קולנוע דרייב-אין.

בעזרת ריבוי-הצורות, יכולה מערכת זמן-ההרצה (הסביבה בה מורצות התוכניות) להבחין בין סוגי עצמים. לצורך דוגמת החניון שלנו, כל מה שנדרש בתוכנית המשתמשת בשיטות כניסה הוא, להצביע על כך שיש צורך בשיטת כניסה. המערכת של זמן ההרצה מוודאת שימוש בשיטת הכניסה הנכונה על ידי זיהוי סוג העצם המבקש את הכניסה, וקריאה לשיטה הנכונה, מתוך אוסף השיטות הזמינות. תוכנית היישומים שלנו צריכה להכיל התייחסות לעובדה שכלי רכב כלשהו נכנס לחניון. עם זאת, היא לא תכיל פרטים בדבר זיהוי הרכב, או מנגנון הכניסה.

תכנות מוכוון-עצמים מאפשר תוכניות פשוטות וקלות יותר לתחזוקה, שאפשר לשוב ולהשתמש בהן ולהרחיבן כדי לענות על דרישות חדשות. **השימוש החוזר** (reuse) חוסך זמן ומאמץ בתיכון, מצמצם את הסיכונים ואת מאמץ הבדיקות. על צד השלילה,

תוכניות מוכוונות-עצמים גדולות יותר בשל ספריות המחלקה השוכנות בהן, והרצתן איטית יותר מאשר תוכנית מסורתית מקבילה, משום שלמערכת זמן-ההרצה יש עבודה רבה יותר לבצע בזמן ההרצה. לדוגמה, זיהוי סוגי עצמים ובחירת השיטות המתאימות מתוך הספרייה, הן משימות תקורה הקשורות בריבוי-הצורות, ואלו חייבות להתבצע בכל שימוש בשיטת ריבוי-הצורות. לא כל השיטות משתמשות בגישת ריבוי-צורות, והתקורה בפועל תשתנה בהתאם, מותנית בגישת התיכון המסוימת שנקבעה.

יתרונות איכות עיקריים של תכנות מוכוון-עצמים

1. רעיון הכימוס או 'אריזת המשאבים' של נתוני העצם ושל הפונקציונליות, פועל כהגנה בפני מספר רב של חולשות בגישות תכנות מוקדמות יותר, בהן היתה לתוכניתנים גישה לכל מקום, ובכל עת.
2. יצירת ממשקים מוגדרים היטב מגדילה את הסיכויים לשוב ולהשתמש בעצם בכל זמן שהוא בעתיד.
3. בניית ספריות של 'מחלקות' (תבניות עצמים) היא אחת הדרכים כדי להשיג את משימת העשרת איכות התוכנה, וצמצום הסיכון הכרוך בפיתוח יישומים.

גישת הפיתוח מוכוון-העצמים

לגישת הפיתוח מוכוון-העצמים שתי תכונות יסודיות:

1. **גישה מבוססת-מוצר (product based)**

גישת הפיתוח המובנית אימצה את תהליך ניתוח הדרישות ויצירת מערכות בצורה המאפשרת למוצרים להופיע בהדרגה. ככלל, המוצר אינו נראה לעין כמעט עד לסוף התהליך. זוהי ***גישה מבוססת תהליך*** (process based approch). **בגישה *מבוססת מוצר*** (product based approch), המוצרים הסופיים מופיעים במועד מוקדם ותוך כדי התהליך משפרים, מרחבים או בונים אותם מתוך רכיבים מוגמרים של השלבים השונים.

2. **גישה לא-קווית (non-linear)**

כפי שהוסבר בפרק 7, במחזור חיים לא-קווי, התהליך פשוט יחסית, ומה שמוליך לבסוף לאספקת המוצר או המוצרים, הוא החזרות על התהליך בוואריאציות קטנות בלבד. פיתוח מוכוון-עצמים כולל גישת כלאיים, המשתמשת בטכניקות איטרטיביות ותוספתיות בתוך תהליך רציף.

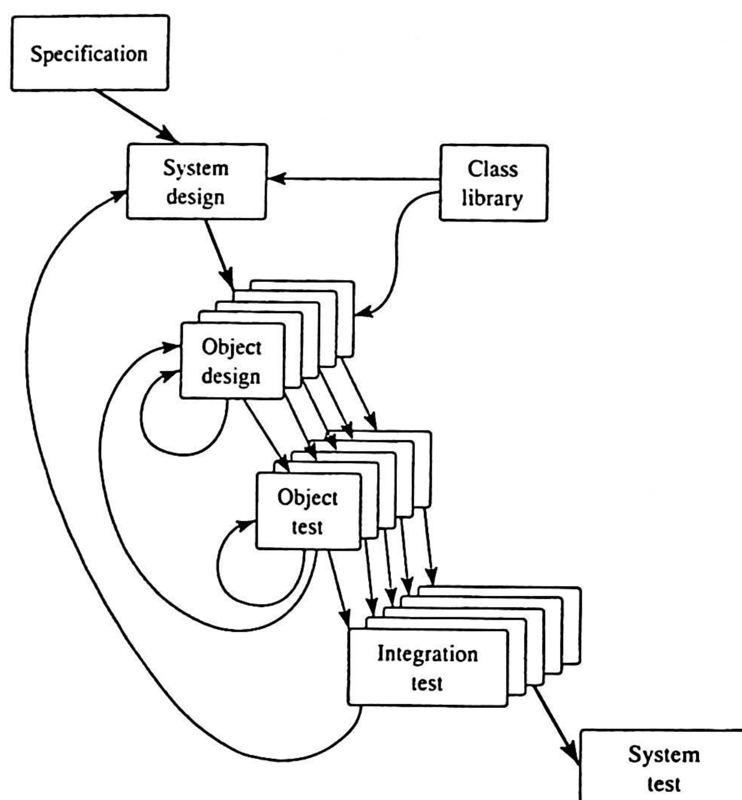
גישת הכלאיים (hybrid)

גישה איטרטיבית היא כמעט לחלוטין גישה החוזרת על עצמה, ותוך כך בונה מודלים שנעשים מפורטים ומדויקים יותר עד שמתקבלת גירסה קבילה שמסופקת למשתמש. בדרך כלל, הגישה התוספתית מקימה ארכיטקטורה ומפצלת אותה למקטעים שניתן לבנותם פחות או יותר עצמאית, ולבסוף מחברת אותם יחד. גישת הכלאיים משתמשת

בשני הרעיונות לסירוגין. היא יוצרת צירוף של גישות רציפות, איטרטיביות, ותוספתיות.

בתרשים 9.4, השלבים ראשון ואחרון נמצאים ברצף עם השלב האמצעי; השלב האמצעי גדול בהרבה מאחרים והוא תוספתי, אך כל תוספת מפותחת באופן איטרטיבי.

פיתוחים מוכווני-עצמים מאמצים לעיתים תכופות הכלאה של סגנונות, האיטרטיבי והתוספתי, אך ניתן להשתמש בהם בגישת פיתוח היישומים המהיר, שנדונה בפרק 8.



תרשים 9.4 פיתוח כלאיים hybrid

יצירת אבטיפוס עם מחלקות

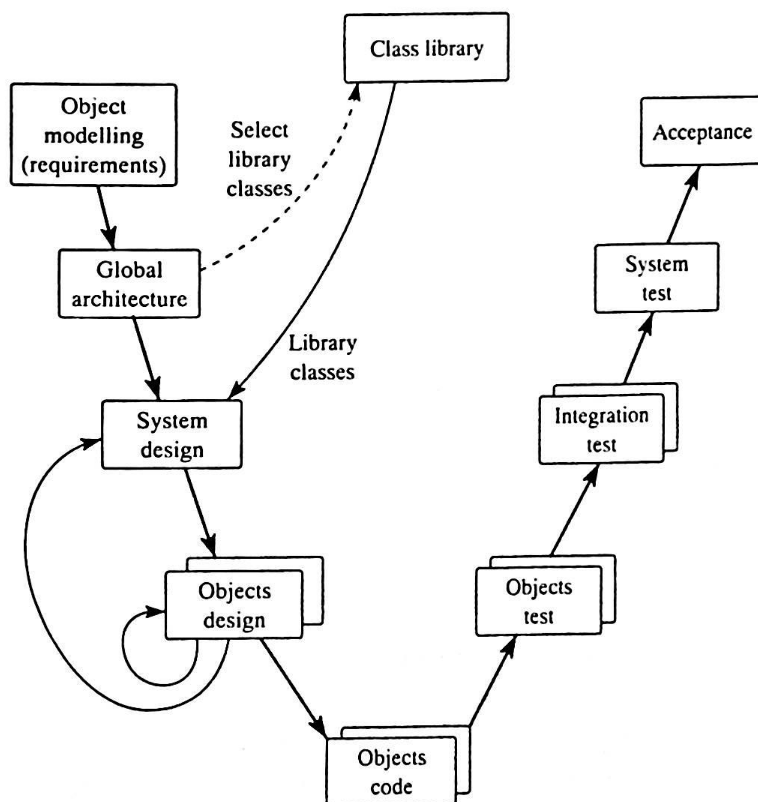
בפיתוח מוכוון-עצמים ניתן ליישם את יצירת האבטיפוס במספר רמות. ברמת המחלקה, ניתן לעצב מחלקות פשוטות בתחילת הפיתוח ולעדכן בהדרגה עם ההתקדמות. ברמת הארכיטקטורה, יישום פשוט יחסית של מודל העצם (object model) יכול להתפתח ולהתרחב כדי ללכוד מאפיינים ותכונות נוספים של עולם המציאות. ברמת המערכת, ניתן להציג מודל עצם פשוט בתחילת הפיתוח ולהרחיבו ככל שמתבררים פרטים נוספים. גישה זו דומה לגישה ההתפתחותית אל הפיתוח.

ניתן היה לאפשר לשלוש השכבות של יצירת אבטיפוס להתנהל בו-זמנית, אך הדבר היה גורר בעקבותיו גישה לא מבוקרת אל הפיתוח. כך היה נמנע, או לא היה קיים כלל ניהול הפרויקט.

כדי שהתהליך יהיה ניתן לשליטה, אנו זקוקים למודל מחזור חיים המנצל את הסגנונות האיטרטיביים והתוספתיים בצורה מבוקרת, כדי שאפשר יהיה להעריך, לתכנן ולשלוט בפרויקט.

מחזור חיים מכוון-עצמים

מחזור החיים החלזוני (ספיראלי) הוצג בפרק 1 כדוגמה של מחזור חיים לא-קווי. בפועל, זהו מודל פתוח מדי שאינו מאפשר את רמת הבקרה לה אנו זקוקים בפרויקטים מציאותיים. דרושה לנו גירסה מתוחמת (constrained) של מודל החלזון או של מודל חלופי אחר, מודל זה יספק מסגרת גמישה לניצול הגישה מוכוונת-העצמים, יספק למנהל הפרויקט מבנה לבסס עליו אומדנים ותוכניות ויאפשר לו את בקרת התהליך.

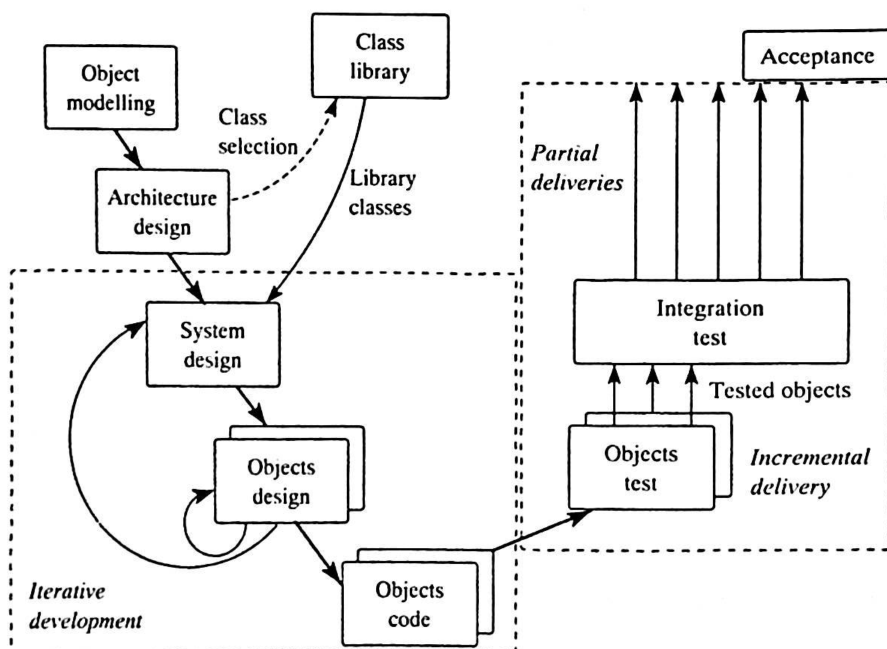


תרשים 9.5 מודל V משופר, עבור גישה מוכוונת-עצמים

אחת הדרכים לטיפול בבעיה היא להתחיל בדבר מוכר שניתן להפעלה בסביבה אחרת, כגון המודל V. ניסיון ראשון לשינוי שיתאים לגישה מוכוונת-העצמים, נראה כמו תרשים 9.5. הוא נאמן למדי לצורה הכוללת של תהליך הפיתוח, אך אי אפשר לבסס עליו מתודולוגיה. הצורה הכללית מזכירה את החילזון, אך עם מחזורים מוגדרים יותר של החזרה. יש בה גם הדים לגישה של תיבת הזמן (timebox). דבר זה מרמז על קביעה כוללת של יעד ועל קבוצת אבני דרך שהגישה אל כל אחת מהן היא באמצעות שיטה איטרטיבית או תוספתית. שלב קביעת יעד, צפוי שיהיה איטרטיבי; הוא מסתיים במודל מְתָאר של עצם, ובשלב זה נעשה חיפוש אחר מחלקות מועמדות.

הצעד הבא יהיה הגדרת ארכיטקטורת המערכת, וחיפוש אחר אסטרטגיית הפיתוח. זו תהיה צירוף כלשהו של התוספתי והאיטרטיבי, עם מדיניות של אבטחת איכות ובדיקות. למשל, נוכל להחליט לבצע מספר בדיקות בכל איטרציה ואחר כך לבצע מבדק קבלה סופי. נוכל גם לבצע ניסוי מערכת כאשר חלק מסוים של התוספות הבודדות יהיה מוכן. אנו גם זקוקים למדיניות סקרים והערכה של מודלים איטרטיביים. יש אסטרטגיות חלופיות רבות, אך כולן כוללות את תכנון האיכות כפעילות בעלת חשיבות קריטית שבאה לפני הפיתוח.

לבסוף, כיצד נסיים את הפרויקט? מה יהיה מנגנון הקבלה הסופי? עלינו לקבוע שיטה להגדרת 'אבטיפוס סופי' ולספקו בשלמות יחד עם אוסף תיעוד חיוני. זה לא יהיה בהכרח תהליך ישיר בסביבה כה נזילה ונעה במהירות, ולכן יש לראות בו מועמד נוסף להגדרה בשלב מוקדם.



תרשים 9.6 מחזור חיים מוכוון-עצמים

הצורה הסופית של תהליך מחזור החיים הבסיסי נוטה במידה רבה לעבר תכנון בתחילת הדרך, שאחריו תבוא סדרת צעדים מוגדרים, עם אפשרות של שינוי הצעדים הספציפיים ככל שהפיתוח נמשך. ולסיום, שלב סופי של המסירה הרשמית. תרשים 9.6 מדגים את הרעיונות הבסיסיים.

ניהול פיתוח מוכוון-העצמים

כיצד נוכל לנהל סביבה דינמית זו של יצירת אבטיפוס כדי להכניס את הפרויקטים למגבלות ההכרחיות של לוח זמנים ותקציב? אין ספק שאנו חייבים לנהל. אחרת, נהיה נתונים לחלוטין בידי כל גחמה נוספת של המפתחים. ברור לא פחות שאיננו יכולים לאכוף מבנה פורמלי וביורוקרטי מאוד לניהול הפרויקט וניהול האיכות. הרי ברצוננו לשמר חלק מהאפשרויות של התיכון החדשני באמת ומוסיף ערך, עבורו נבנו השיטות מוכוונות-העצמים.

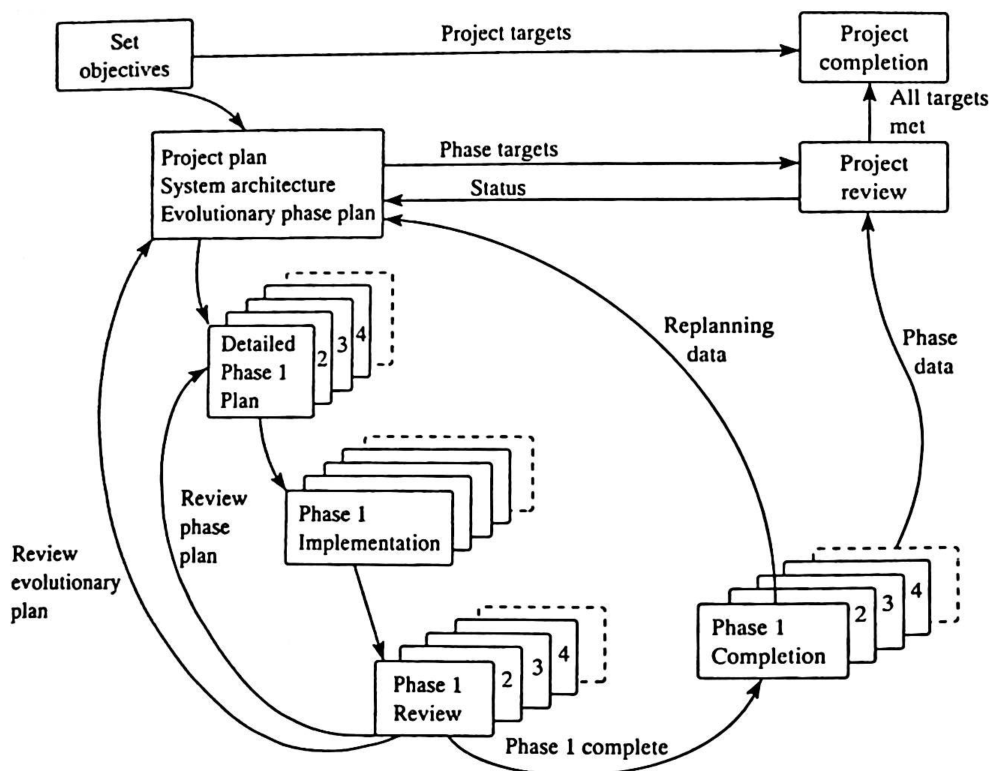
הפשרה המתבקשת היא: שלב תכנון זהיר ומפורט המקדים את הפיתוח, בו נקבעים כללי יסוד ברורים ביותר להערכת ההתקדמות ולשינויי התוכניות. מעת שיותחל בפיתוח, יקבל הפרויקט תאוצה משלו ולתהליכים בלתי מוגדרים תהיה נטייה להישאר בלתי מוגדרים. איננו יכולים להרשות לעצמנו את המותרות שבתהליכים בלתי מוגדרים ואת התוצאה של תנועה חופשית במדרון שנגרמת בגינם. לא נוכל לאפשר מצב של חוסר הגדרות בתוכניותינו, ולכן עלינו להבטיח לעצמנו יכולת לנהל בעקביות את משאבי הפרויקט לעבר היעדים ארוכי הטווח.

הגישה שננקוט דומה באופן כללי לגישה של תיבת הזמן שתוארה בפרק 8, אלא שהפעם יש לנו פחות אי-ביטחון בתוצאות. נשתמש בשיטות איטרטיביות ותוספתיות כדי לאפשר גישת תיכון ישירה יותר, אך בשלבים קלים יחסית. הגישה הישירה מסייעת לנו לפקוח עין על היעד, בעוד השלבים הקלים גורמים לתהליך להיות נתון יותר לניהול.

כמו בשיטות של יצירת אבטיפוס, אנו פותחים ביעד כולל וב'ארכיטקטורת פרויקט' המזהה סדרת שלבי פיתוח ויעדיהם (הארעיים). אפשר לתכנן בפירוט את השלב הראשון, בו תוגדר ארכיטקטורת המערכת ותזוהינה המחלקות המועמדות. שלבים עתידיים יפתחו איזו שהיא קבוצת משנה של הארכיטקטורה הכוללת, על ידי צירוף מחלקות שנבחרו מתוך הספרייה, הגדרה ובנייה של מחלקות חדשות, או שילוב כלשהו של שתי הדרכים.

בכל שלב פיתוח, חייב להתבצע סקר של התוכנית כדי לקבוע:

- אם השלב שזה עתה הושלם השיג את יעדי התיכון שלו;
- אם יש צורך להתאים בצורה כלשהי את יעדי השלב הבא;
- אם השלבים שהוגדרו מלכתחילה הם עדיין השלבים הנכונים, ואם יעדיהם מוגדרים כהלכה;
- אם היעדים הכוללים הם עדיין ברי-תכנות; במקרה שלא, מה צריכים להיות היעדים המתוקנים.



תרשים 9.7 ניהול פרויקט עבור פרויקטים מוכווני-עצמים

במסגרת כל שלב, ניתן להגדיר וליישם את עבודת הפיתוח הממשית בצורה קווית, או שניתן לטפל בה בגישת האבטיפוס. הבחירה תותנה בעבודה בה נעסוק באותו שלב, שעשויה להיות פחות או יותר מוגדרת כבר. מתודולוגיה זו לניהול פרויקטים מתוארת בתרשים 9.7.

סקירה מתמדת של יעדים ברורים ומכומתים היא, כמו בכל הפיתוחים הלא-קוויים, בעלת חשיבות יסודית. בקרה של פעילויות הפיתוח היא לא פחות קריטית, וחייבת להיות נתונה לאותן דיסציפלינות כמו כל גישה של שימוש באבטיפוס. תכנון האיכות הוא סוגיה נכבדה בשל רמת הסיכון. אפילו המתודולוגיה הבסיסית אינה נחלת הכלל עדיין, והכישורים הנדרשים עבורה גם הם אינם כה נפוצים.

מהן סוגיות האיכות

מהן סוגיות האיכות שמעוררות שיטות מוכוונות-עצמים? הסוגיות רבות, אך מתחלקות לקבוצות נבדלות. להבנת הסוגיות המפורטות חיוני שתהיה לנו תפיסה מתאימה של סוגיות מדיניות האיכות.

12 סוגיות חשובות בנושא מדיניות האיכות לפיתוח מוכוון-עצמים

1. האם כל המעורבים מבינים את המינוח, והאם הבנה זאת אחידה לגבי כולם?
2. האם עומדים לרשותנו שיטה הולמת וסביבה וכלים לפיתוח, והאם קיבל כל הסגל הרלבנטי הדרכה מתאימה?
3. האם משמעויות הטכנולוגיה מובנות להנהלה?
4. האם יש לנו מתודולוגיה לניהול פרויקטים לצורך ניהול אמין של הפרויקטים האלה?
5. האם אנו יודעים כיצד לנסות מודלים מוכווני-עצמים? האם זמינים לנו כלים מתאימים וטכניקות הולמות, שיאפשרו להגדיר אסטרטגיית בדיקות?
6. האם יש ברשותנו מומחיות במספר מספיק של תחומי הארגון, שתאפשר לנו לבצע אפיון אפקטיבי וסקרי התיכון?
7. האם אנו מבינים את ההשלכה של הטכנולוגיה החדשה על המערכות המסורתיות שלנו (legacy system)?
8. האם התוכניות מוכוונות-העצמים החדשות תפעלנה יחד עם מסדי הנתונים (הטבלאיים) הקיימים שלנו?
9. האם העובדים בסגל האיכות הקיים שלנו מבינים את הטכנולוגיה, והאם הם פיתחו כבר בעבר מתודולוגיית איכות עבור מערכות מוכוונות-עצמים?
10. האם שקלנו את הדרכים למדידת הביצועים של התהליך והמוצרים החדשים מנקודת ראות של שיפורים לטווח ארוך?
11. האם כל חלקי הארגון מסכימים להגדרת היעדים? במקרה שלא, האם יש ברצונם להפעילם בצורה שונה?
12. האם תכננו לשלב את כל היישומים מוכווני-העצמים שלנו במועד כלשהו בעתיד? אם כן, האם שקלנו כיצד ינוהל הדבר ועל ידי מי?

בהיותנו מצוידים בתובנות אלו של המדיניות, נוכל להתפנות אל היבטים ספציפיים יותר של הטכנולוגיה ושל ניהולה.

1. **סוגיות טכנולוגיות**

סוגיות אלו מתייחסות לסיכונים הקשורים בשימוש בטכנולוגיה חדשה ובלתי מנוסה יחסית. הסוגיות כוללות:

- ◊ המפתחים חייבים לקבל הדרכה מתאימה כדי להכינם לעבודה שלפניהם;
- ◊ חייבים להיות לנו מחזור חיים ומתודולוגיה מוגדרים היטב;
- ◊ חייבים להיות לנו תקנים ונהלים עבור סביבת הפיתוח החדשה;
- ◊ חייבת להיות לנו קבוצת כלים מתאימה;
- ◊ אנחנו חייבים להיות מסוגלים לבחון יעדים ומערכות מונחות יעדים.

2. **סוגיות של ניהול פרויקטים**

סוגיות אלו מתייחסות לסיכונים שבניהול פרויקטים על פי תהליך פיתוח חדש. הסוגיות כוללות:

- ◊ חייבת להיות מתודולוגיה לניהול פרויקטים שתהיה מבוססת על תהליך הפיתוח המוסכם;
- ◊ חייבת להיות שיטת תכנון עבור פיתוחי כלאיים;
- ◊ חייבים להיות מסוגלים להעריך בדיוק את הזמן והעלות של פעילות פיתוח מוכוונת-עצמים;
- ◊ חייבת להיות לנו שיטה לסקירת פעילויות הפיתוח כדי לקבוע אם הן יכולות להסתיים בהצלחה ולספק תוצאות קבילות;
- ◊ חייבים להיות לנו אמצעים לניטור ההתקדמות בשלבים האיטרטיביים של הפיתוח;
- ◊ חייבים להיות מסוגלים לחזות את המאמץ והזמן הדרושים עד להשלמה, לגבי פרויקטים שהושלמו רק בחלקם.

3. **סוגיות של ניהול האיכות**

סוגיות אלו מתייחסות למידת הבנתנו את תהליך הפיתוח שביסוד הדברים, ואת דרך ניהולו. הסוגיות כוללות:

- ◊ חייבת להיות לנו מדיניות ברורה לדרך הטיפול בטכנולוגיה חדשה זו. אחדות מסוגיות המדיניות החשובות ביותר כבר נדונו;
- ◊ חייבת להיות לנו הגדרה ברורה של תהליך הפיתוח, כדי שנוכל לעקוב אחריו ביעילות;
- ◊ חייבת להיות לנו אסטרטגיה של אימות ובדיקת תקפות עבור פרויקטים מוכווני-עצמים;
- ◊ חייבים להיות לנו מדדים שימושיים למוצר ולתהליך, כדי שנוכל להמשיך ולשפר את שיטותינו.

מספר בעיות איכות מעשיות

הקדשת תשומת לב לסוגיות האיכות שמנינו, תסייע בהספקת מסגרת עבור הפיתוח מוכוון-העצמים, אך ניהול הטכנולוגיה עדיין נותן פתח למספר בעיות מעשיות קשות. יש עוד בעיות אחרות הקשורות בטכנולוגיה עצמה. אחדות מהבעיות הדחופות והחשובות ביותר הן:

1. **ניהול התצורה**

כל טכניקה שכרוכים בה עדכונים חוזרים של מודל או של אבטיפוס מביאה בעקבותיה סיכונים הכרוכים בבקרת שינויים. דאגתנו הסופית תהייה שתימסר הגירסה הנכונה, אלא שאנו עלולים לגרום לבעיות פיתוח חמורות ביותר אם נאפשר לריבוי פעולות פיתוח להימשך במקביל ללא בקרה מתאימה. פיתוח

מוכוון-עצמים מגביר את הבעיה משום שקיימות בו רמות אחדות של מודלים, וכל רמה נתונה לשינויים שעשויים להשפיע על הרמות האחרות.

2. **ספריות מחלקה**

רבים מהיתרונות של פיתוח מוכוון-עצמים מחייבים יצירה וניהול ספריית מחלקה (class library) שתשמש כמסד מידע (repository) להגדרות העצמים שנבנו ונבדקו כבר, ויכולים להביא תועלת בפיתוחים עתידיים. ספרייה כזו מחייבת בקרת הגישה וניהול רשומות, ממש כמו כל ספריית פרויקט אחרת. יש גם סוגיות אחרות הקשורות בניהול הספרייה, כגון בחירת תוספות, הגדרה ואכיפה של תקנים לעצמים המופקדים בספרייה, וניהול מעקב אחר המקומות בהם משתמשים בעצמי הספרייה.

3. **הגדרות עצמים**

מערכת מידע כלל-מפעלית רבת עוצמה מחייבת שכל היישומים יתפעלו את אותם העצמים, כך שייעשה **שימוש חוזר** בעצמים של הספרייה בכל הארגון, וכל היישומים יכילו ממשקים אל עצמים עסקיים סטנדרטיים. כדי שזה יהיה המצב, יש צורך בהסכמת כל המשתמשים להגדרה של כל עצם. כשיש לעצמים הגדרות פשוטות מאוד אין ההסכמה מהווה בעיה, אך הספרייה מתמלאת במספר גדול מאוד של עצמים פשוטים ואובדים יתרונות השימוש החוזר, משום שהיישומים נעשים מורכבים כמו מערכות שאינן מוכוונות-עצמים. ככל שהעצמים מועשרים בתוכן עסקי, כן גדלים הסיכויים לאי-הסכמה נרחבת ביחס לפרטים.

4. **מהימנות העצמים**

עצמים בתוך ספרייה חייבים להיות אמינים מאוד ומוגדרים היטב, משום שהם עתידים להוות את 'ליבם' של יישומים מסוגים שונים. איכות הפיתוח של עצמי הספרייה חייבת להיות בעלת חשיבות עליונה, גם כאשר בונים את העצמים בתהליך של יצירת אבטיפוס איטרטיבי.

5. **מסדי נתונים טבלאיים**

יישומים מוכווני-עצמים רבים יידרשו לפעול עם מערכות קיימות של ניהול מסדי נתונים טבלאיים. דבר זה יהיה כרוך בבניית ממשקים מתאימים בין הטכנולוגיות השונות זו מזו, יגדיל את המורכבות ביישומים ויקטין את הגמישות והמהימנות.

6. **בחירת עצמים**

בכל פעם ששוקלים התחלת פרויקט חדש, צריך להחליט אם לעשות שימוש חוזר בעצמים קיימים, או לבנות עצמים חדשים. הרצון להפיק את מירב היתרונות מהשימוש החוזר מחולל לחץ להשתמש בו, אלא שדבר זה עלול להתבטא בצורך לבנות תוכנות יישומים חדשות כדי למלא את הפערים שיווצרו על ידי השימוש החוזר בעצמים קיימים, שלא תמיד עונים על הדרישות במדויק. התוצאה עלולה להיות הגברת המורכבות.

תכנון האיכות

האם אמנם יש בגישה לא-קווית מסוימת זו יכולת להתאים את עצמה לגי
האיכות שתקן ISO 9000-3 מעדיף? אין סיבה שלא, אם כי יש צורך בפרשנ
ויש נושאים שצריך לטפל בהם, שלא כוסו ברשימת התכנים שהומלצה על יד
המנחים.

תכנון איכות עבור פיתוח מוכוון-עצמים

תוכנית איכות עבור פיתוח מוכוון-עצמים אמורה לכסות את כל סוגיות
הקריטיות:

1. **ספריות מחלקה**

   מהן ספריות המחלקה הזמינות? מי ינהל את ספריית המחלקה של הפרו
   יחליט על הארכיטקטורה ומי יבחר את מחלקות הספרייה?

2. **ניהול התצורה**

   באיזה תהליך בקרת שינויים נשתמש וכיצד יזוהו המודלים?

3. **בדיקות**

   מה תהיה אסטרטגיית הבדיקות הכוללת? באילו טכניקות נשתמש? כיצ
   הבדיקותניסויים בעצמים הבודדים (לצורך הכללתם בספרייה)?

4. **קבלה**

   מהם הקריטריונים לקבלה ובאיזו דרך יודגמו? מה יהיה מנגנון המס
   המוצר?

5. **תכנון הפרויקט**

   מה תהיה צורת התוכניות? מה תהיה שכיחות עדכונן?

6. **הגדרת עצמים**

   מי יהיה אחראי להשגת התאמה בין עצמים עסקיים לבין המשתמשים והס

7. **סביבה**

   באיזה שיטות וכלים נשתמש?

8. **בקרה**

   מהם הקריטריונים שיגדירו את השלמת שלבי הפרויקט? כיצד יינתן
   להמשיך לשלב המידול הבא, ועל ידי מי?

9. **תיכון עצמים**

   מהם הכללים שישלטו בתיכון העצמים במונחים של ממשקים ציבוריי
   ההכללה? מי יחליט על הרמה ההולמת עבור כל אחד מהעצמים?

10. **סוגיות בפרויקטים**

    יש צורך לטפל בכל הסוגיות 'הסטנדרטיות' שתוכנית איכות מטפלת בהן.

האם פיתוח מוכוון-העצמים הוא אמנם התשובה הנכונה?

פיתוח מוכוון-העצמים מייצג התקדמות טבעית משיטות פיתוח מבניות שהיו מקובלות משנות ה-80. במבט ראשון נראית שיטת פיתוח זו כמהפכנית למדי ושונה מכל קודמותיה. עיון מעמיק יותר יגלה, כי היא מהווה הרחבה של עקרונות המבניות שנחשבו כאבני יסוד להנדסת תוכנה. עם זאת, שיטה זו קוסמת לאלה המעדיפים גישה בלתי פורמלית יחסית לפיתוח, משום שהיא מעודדת את סגנון יצירת אבטיפוס.

מה בנוגע לנקודת המבט של האיכות? גם בתחום זה, יש תכונות המוסיפות על יתרונות השיטות התבניתיות המוקדמות יותר; בעיקר, ההורשה והשימוש החוזר, שמצמצמים את היקף הפיתוח החדש ואת השפעת השינויים על תוכנה קיימת. שני גורמים אלה היוו בעבר תחומי סיכון משמעותיים לגבי טכנולוגיות התוכנה. עם זאת, העובדה שהטכנולוגיה צעירה יחסית ומחייבת שימוש בכישורי תיכון חדשים, מחוללת סיכונים חדשים. אלה ייעלמו במרוצת הזמן, אך מהווים לפי שעה מגבלה רצינית לכל מי ששוקל פיתוח מוכוון-עצמים לראשונה.

מנקודת ראות של ההנהלה, פיתוח מוכוון-עצמים יוצר בעיות חדשות עבור מנהל הפרויקט ומציע הזדמנויות חדשות למנהלים בתחומים העסקיים. סוג חדש של מחזור חיים, עם קווי דמיון למחזורי החיים בסגנון V והחילזון; מתודולוגיות הכוללות פיתוחים בסגנון רציף, איטרטיבי, ותוספתי; מבני תוכניות שיוצרים אתגרי בדיקה חדשים; כל אלה מצטרפים לכלל מבחן חמור להנהלה וסביבה מחמירה מאוד בכל הנוגע להספקת תוצאות בזמן ובמסגרת התקציב.

אם כן, האם הפיתוח מוכוון-העצמים מהווה אמנם את התשובה הנכונה? ניסיון העבר מרמז שיש לנקוט זהירות רבה בכל פעם שמתעוררת שאלה ממין זה. עם זאת, אין ספק שפיתוח מוכוון-עצמים הוא צעד בכיוון הנכון. ימים יגידו לאן הוא יהיה מסוגל להביא אותנו, אך כבר עכשיו ברור שאין זה פתרון רגעי לבעיה הוותיקה של ניהול פיתוח התוכנה. עלינו ללמוד לרתום כלי חדש זה ולהשתמש בו באפקטיביות לפני שנוכל להעריך את התרומה שהוא יכול לתרום.

לאלה שטרם עשו את 'הטבילה הראשונה' בפיתוח מוכוון-העצמים, לפנינו מספר נקודות ציון:

עשרה צעדים לעוסק בפיתוח מוכוון-עצמים

1. **הדרכה לכל**

דאג להדרכת אנשי המפתח בהקדם האפשרי. לא רק המפתחים: אנשי האיכות, משתמשים ומנהלים צריכים להבין את העקרונות ושפת היומיום המקצועית.

2. **צור מספר תקנים**

קבל את הסכמת המפתחים והעוסקים באיכות במספר נושאי מפתח טכניים.

3. **תכנן את דרך התכנון**

לפני התחלת העבודה, קבע את מבנה הפרויקטים ומה תהיה צורת התוכניות.

4. **החלט כיצד לבצע בדיקות**

בעבודתך תזדקק למספר טכניקות וסטנדרטים; שקול את הדרך בה תוודא את תקִפות התוכנה לפני שאתה מתחיל בכתיבתה.

5. **מכור להנהלה את מה שהכנת**

וודא שההנהלה שלך מודעת לסיכונים וכי היא עומדת מאחוריך.

6. **דאג להתעניינות המשתמשים**

הכנס את המשתמשים לתוך התהליך, כדי שיוכלו לראות את היתרונות ולהעריך את הקשיים.

7. **תרגל את הטכניקות לפני השימוש בהן**

השתמש בפעילויות בשיטת הסדנה, כדי לעבור על צעדים פשוטים לפני שתשתמש בטכנולוגיה ברצינות. דאג שכולם יהיו מעורבים בסדנאות.

8. **הקם את הספרייה שלך**

צור ספריית מחלקה, עבד את דרך השליטה בה, ומנה את 'הממונה על המחלקות'.

9. **דאג למבנה נכון של הצוות**

החלט מהם הכישורים הדרושים בצוות הפיתוח, ובחר בקפדנות את האנשים שיאיישו אותו. ראה צוות זה כמי שיהיה במרכז תוכניותיך העתידיות.

10. **עכשיו אתה מוכן לפרויקט-חלוץ פשוט**

כל ההכנות האלו הביאו אותך לנקודה בה אתה יכול כבר לשקול מעבר לעבודה ממשית. עם זאת, וודא שהפרויקט יהיה קטן ולא קריטי מדי.

לאחר שתתחיל במתודולוגיית הפיתוח שלך, תוכל לחזור אל רשימה זו ולעדכנה, או להכין רשימה משלך. מומלץ לעשות זאת, משום שתרגיל מסוג זה הוא בדיוק הדבר שמניע אותנו להרהר בסוגיות החשובות באמת, ומסייע לנו להימנע מתלות משעבדת בשיטות ובגישות שיש צורך להתאימן להתנסות שלנו.

חלק 5

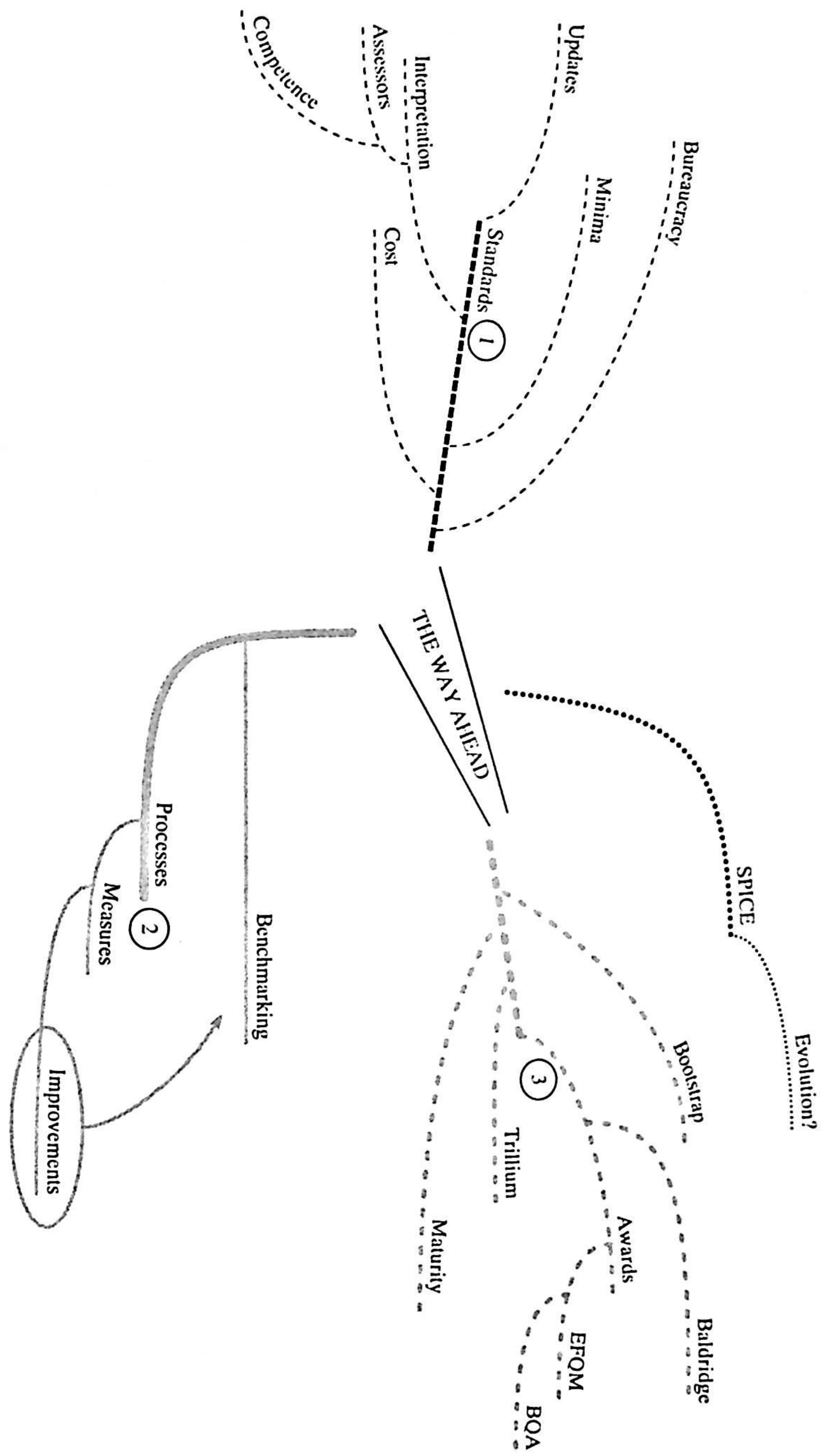
עתיד איכות התוכנה

חלק 5 עוסק בהתבוננות אל העתיד. הוא מכיל פרק אחד בלבד - פרק 10: הדרך שלפנינו, מה שמעבר לתקן ISO 9000-3. מטרת פרק 10 היא לשמש כעין 'אחרית דבר' ולהבליט מספר רעיונות מפתח שיש לבוחנם למען העתיד.

הספר זיהה עוצמות וחולשות בגישת התקנים ISO 9001 ו- ISO 9000-3. הוא גם עסק בגישות אחרות לנושא ניהול איכות. פרק 10 מציב את הגישות החלופיות במסגרת הקשר זה, ומזהה את המגבלות הפוטנציאליות המשמעותיות ביותר, שיש בחלקן או בכולן. בהסתמך על טיעונים שהושמעו בחלקיו השונים, מחפש הספר אחר אסטרטגיה ההולכת ומתהווה ואשר תשרת את צרכיה של תעשיה צעירה ומתפתחת.

רעיונות המפתח והקשרים שביניהם מוצגים במפת התפיסה הכללית שבתרשים ח.5.1.





תרשים ח.1.5 מפת התפיסה הכללית

פרק 10

הדרך שלפנינו

טכנולוגיית המידע (IT) היא המשרת של התעשיות המשתמשות בה כדי לנהל ולהעשיר את פעולותיהן. בכל דרך בה תשתנינה תעשיות אלו בעתיד, שביעות הרצון של הלקוחות תספק בצורה זו או אחרת את ההנחיות, בעיקר את ההנחייה השימושית ביותר בתחום האיכות. האם יוכל תקן ISO 9000-3 להמשיך ולהשפיע אפקטיבית בסביבה ההולכת ומתפתחת? איזו גישה של ניהול האיכות תשרת את שיטות הפיתוח החדשות, ההולכות ומתגלות לעינינו? מהי האסטרטגיה ארוכת הטווח עליה נוכל להמליץ, אם בכלל?

האם אנו זקוקים לדרך קדימה?

לפני שאנו מפליגים לחיפוש רציני אחר איכות תוכנה, עלינו לשכנע את עצמנו שיש אמנם צורך בחיפוש כזה. האם איכות התוכנה גרועה עדיין עד כדי כך שכל העת עלינו לחפש דרכים חדשניות יותר כדי לקדמה? התשובה חיובית באופן חד-משמעי. ההתקדמות במהלך למעלה מ- 40 שנה של פיתוח תוכנה היתה צנועה למדי במונחי האיכות, וגם אם הטכנולוגיה מסנוורת, האיכות היא שקובעת בסופו של דבר את קבילות מוצרי תוכנה.

ואם עלינו להמשיך בדרך האיכות, לאן עלינו לפנות? כנקודת מוצא, נראה שמן הראוי לבחון תחילה את היוזמות הקיימות. מהן אפוא האפשרויות הפתוחות לפנינו? קיימות היום ארבע גישות של המגמות הראשיות:

- פיתוח **תקנים של מערכת ניהול איכות**, בהן כבר דנו;
- גישת **בשלות התהליך** (process maturity) המגדירה רמות בשלות כמטרות לשיפורים;
- גישת **ההערכה העצמית** (self-assessment), המספקת מודלים שניתן לערוך מולם הערכות מכומתות;
- גישת **שיפור התהליכים** (process improvement), המעודדת מידול תהליכים ושיפור-תהליכים המבוסס על מדידות.

נסקור בקצרה כל אחת מארבע גישות אלו.

הגישה המבוססת על תקנים

כבר בחנו גישה זו בפירוט מסוים וזיהינו אחדות מחולשותיה, לכן נוכל להגביל עצמנו כאן לסיכום בלבד. מהן החולשות העיקריות?

- מערכות ניהול איכות (QMS) יכולות להפוך לביורוקרטיות מאוד, וקשה לעצור את הסחף בכיוון זה;
- מערכות ניהול איכות מגדירות רק את המינימום הקביל של רמת ההישגים, ואינן קובעות יעדים שיש בהם אתגר;
- סכימות הרישום מעודדות את הארגונים לשמור על המצב הקיים, במקום לחפש אחר שיפורים מתמידים;
- הפרשנות שניתנת לתקנים אינה תמיד חד-משמעית;
- סכימות הרישום מהוות תקורה גבוהה עבור מבצעיה. בעיקר אם מדובר בחברות קטנות יחסית בעלות מנגנון קטן וארגון פשוט.
- גם כאשר בוחרים בזהירות את המעריכים, לא תמיד הם מבינים בצורה מספקת את התעשיות בהן חברים הארגונים שהם נקראים להעריך.

בבריטניה, סכימת TickIT - סכימה סקטוריאלית עבור תעשיית התוכנה, ניסתה לטפל באחדות מסוגיות אלו, בעיקר בסוגיה האחרונה. סכימת TickIT הצליחה מאוד במשיכת חברות תוכנה לבקש הסמכה (certification), אך היא סובלת בכל זאת ממספר פגמים שאת חלקם לפחות ניתן ליחס למגבלות תקן ISO 9000-3.

כיום, לאחר שתקן ISO 9000-3 נמצא בשימוש תקופה ארוכה למדי שיש בה כבר כדי להשפיע השפעה אמיתית, מתעורר הצורך לשקול את העתיד. האם ימשיך תקן ISO 9000-3 לשקף את הנוהג הטוב יותר בטכנולוגיית המידע (IT) בעשור הבא? ככל שמתפתחות התעשיות שטכנולוגיית המידע משרתת ומשתנים צורכיהן, כן חייבת להשתנות טכנולוגיית המידע שמשרתת אותן. הצבענו כבר על אחדות מהמגבלות הפוטנציאליות של תקן ISO 9000-3, וכאן ננסה לחבר את קצות החוטים ולהסיק מספר מסקנות.

בעיית מחזור החיים

תקן ISO 9000-3 אינו מחייב שימוש במחזור חיים מסוים דווקא. אחד היתרונות של תקן זה הוא שימת דגש על גישת מחזור החיים, אך מבלי להכתיב דבר. עם זאת, ממבנהו וצורת הצגתו משתמע מודל מסוים של מחזור החיים שקראנו לו בשם **מחזורי חיים קוויים** (linear life cycles).

תקן ISO 9000-3 מזכיר במקומות שונים את עניין שלבי הפיתוח. בתקן מודגש שהשלבים האלה אינם מתייחסים לשלבי מודל, או מתודולוגיה מסוימים של מחזור חיים. עם זאת, תיאורי גישת השלבים כפי שהיא מומלצת בתקן זה כרוכה בסדרת שלבים, כשפלט שלב אחד משמש קלט לבא אחריו. מכאן משתמעת בהחלט גישה קווית של שלב אחר שלב, לנושא פיתוח התוכנה.

לא נטען שתקן ISO 9000-3 אינו מכיר באופי האיטרטיבי של תהליך הפיתוח. עם זאת, ההנחה היא שהשלבים הם חלק מרצף פעילויות הולך ונמשך. האיטרציה כרוכה בחזרה אל שלב מוקדם, וחזרה עליו, במקום להתקדם. זו גישה השונה ביסודה מגישה שמצפה שהתוכנה תתפתח דרך איטרציות של מערכת הבאות זו אחר זו, כאשר כל איטרציה מתקרבת אל המטרה הסופית. קראנו לגישה זו בשם **שיטת פיתוח לא-קווית** (non-linear development method).

גם לא נטען כאן שאי אפשר להשתמש בתקן ISO 9000-3 בסביבת פיתוח תוכנה המבוססת בעיקר על שיטות לא-קוויות. העקרונות שאומצו בקווים המנחים של תקן ISO 9000-3 ישימים וניתנים לשימוש במגוון מצבים, אך השפעתם תהיה גדולה יותר במקומות בהם קל לפרש וליישם את הקווים המנחים כמות שהם.

ניהול פרויקטים לא-קוויים

כפי שנראה, פיתוח תוכנה מתרחק ממתודולוגיות קוויות ו'מובנות' אל גישות לא-קוויות, הכרוכות בשימוש בשיטות אבטיפוס ומוכוונות-עצמים. האם תקן ISO 9000-3 מספק הנחיות מתאימות עבור מפתחים המשתמשים בשיטות איטרטיביות ועבור מנהלי פרויקטים שמנהלים ושולטים בפרויקטים לא-קוויים? במקרה ולא, תלך ותגבר אי-הרלוונטיות של תקן זה, והתוצאה עלולה להיות חזרה אל שיטות הפיתוח 'אד-הוק' שקדמו לו.

איננו צריכים להניח ששיטות לא-קוויות מייצגות בהכרח את עתיד הנדסת התוכנה. לעומת זאת, נראה שיהיה מסוכן יותר להניח ששיטות מובנות קוויות ימשיכו לשלוט בזרם הראשי של פיתוח התוכנה, אם אמנם היו אי פעם במעמד זה. לעדכון תקן ISO 9000-3, כדי להביאו לשקף את השיטות הלא-קוויות, יכול להיות ערך בטווח הקצר. הסוגיה העיקרית היא, האם ניתן לשמור על רמת עדכון התקנים והקווים המנחים הנלווים להם, כך שתשקף את מגמות הטכנולוגיה הנוכחיות. הסיכון גבוה ביותר לגבי המשתמשים בטכנולוגיות החדשות, וזו גם הקבוצה שתקבל את התמיכה המועטה ביותר מגישה זו.

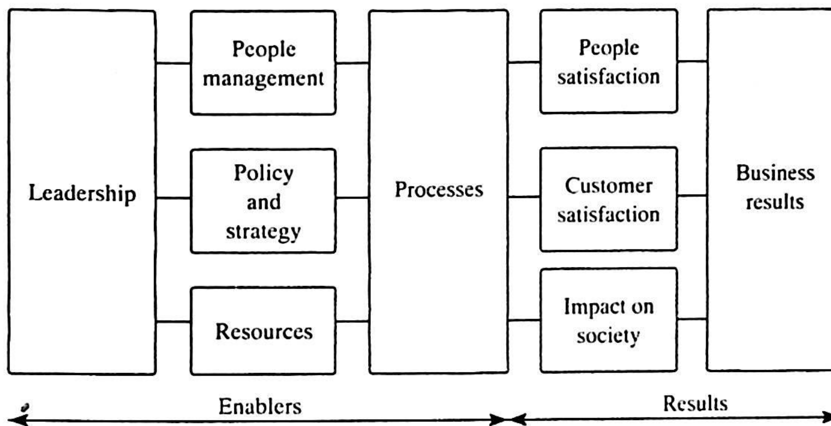
מיומנות הבוחנים

בעיה נוספת הקשורה בגישה המבוססת על תקנים, היא הצורך במעריכים בעלי יכולת טכנית, שרכשו ניסיון בתעשיה בה מתבצעת ההערכה. עד עתה מתקבל הרושם שהסכימה הבריטית של TickIT פתרה בהצלחה את הבעיה ההתחלתית של גיוס והדרכת קבוצת מעריכים. עם זאת, נותרה עדיין בעיית תחזוקת היכולת הטכנית של המעריכים. בהנחה שהתוכנה היא תעשיה הנתונה לשינויים מהירים, בה ההסמכה ניתנת לשלוש שנים בלבד ולאחריהן יש כבר צורך בהערכה חוזרת, כיצד יוכל מעריך שעוסק בהערכות במשרה מלאה לרכוש את הניסיון הדרוש בטכנולוגיות המתפתחות, כדי לשמר את האמינות שלו? הדרכה יכולה לסייע, אך יש הכרח גם בניסיון מעשי כדי להיות מסוגל להבין לאשורן את הבעיות שעומדות בפני המפתחים.

הגישה המבוססת על מודלים

מודלים לאיכות וסכימות הערכה

יותר ויותר אנו רואים סכימות הערכה המבוססות על איכות טוטלית (Total Quality - TQ), כגון הערכת מלקולם באלדרידג׳ בארה״ב והערכות דמינג ביפן. הגישה נשענת במידה רבה על הערכה עצמית (self-assessment), ונתמכת על ידי הערכה רשמית ובמידת הצורך גם יותר בלתי תלויה, כבסיס לקביעת יכולת החברה. אם כי המודלים שמשמשים לסכימות השונות ייחודיים כולם, יש קווי דמיון מרשימים ביניהם. יש כמובן גם קווי דמיון ברעיון של הערכה בלתי תלויה מול תקן כדוגמת ISO 9001. המודל האירופי שמשמש להערכת האיכות מוצג בתרשים 10.1 ומתואר בתור דוגמה (Hakes, 1994).



תרשים 10.1 המודל האירופי להערכת האיכות

המודל מזהה תשעה גורמים שאובחנו כתורמים לאיכות. כל גורם מקבל ציון אינדיבידואלי. הגורמים האינדיבידואליים מכונסים לשני תחומים, תחום המייצג את הגורמים שמאפשרים את האיכות (כיצד מנוהלות פעילויות האיכות של הארגון ומובאות לאופטימיזציה), ותחום המודד את השיפורים העסקיים בפועל. לשתי הקבוצות משקל שווה במודל. המודל מאגד לא רק את האמצעים להשגת איכות, הוא גם מודד את הביצועים התפעוליים והכספיים ואת ההשפעה שיש לארגון על החברה והסביבה.

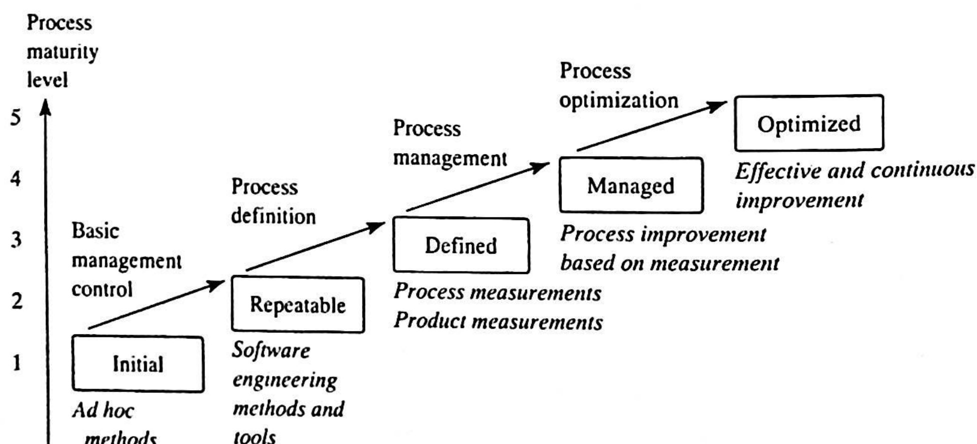
לכל אחד מהגורמים שבמודל יש תיאור נלווה המתאר את מה שהגורם מבקש למדוד, וכיצד ניתן להעריך את הארגון מול גורם זה. מעריכים מיומנים יכולים לבצע השוואות אובייקטיביות בין ארגונים על סמך גורמים אלה, וסכימת המודל האירופי להערכת האיכות משתמשת בהערכות כאלה לבחירת הזוכים.

אפשר להשתמש במודל גם לצורך הערכה עצמית, על ידי הדרכת מעריכים מתוך הארגון. זאת הופכת להיות שיטה משמעותית להשגת שיפורים מתמידים, במסגרת שמאפשרת השוואות מול ארגונים אחרים, ללא צורך בתחרות על הערכה חיצונית.

המודל הישראלי לניהול איכות בתעשיה מפורט בנספח.

בשלות התהליך והערכת היכולת

תחילתה של 'תנועתי' בשלות התהליך היתה למעשה, בשנות ה-80, כאשר המכון להנדסת התוכנה של אוניברסיטת קרנגי-מֶלון פיתח עבור מחלקת ההגנה של ארה"ב שיטה להערכת תהליך התוכנה ומודל בשלות היכולת (capability maturity model - CMM). המטרה המקורית היתה לספק אמצעים להערכת קבלני פיתוח התוכנה. המודל הבסיסי שביסוד ההערכות מכיר בחמש רמות של בשלות. תרשים 10.2 מתווה את המודל. סכימות אחרות, כגון 'ניתוח האיכות והפריון של התוכנה' וה'שיטה להערכת תוכנה' של בריטיש טלקום, אימצו גישה דומה, אך מותאמת לתרבותם המסוימת.



תרשים 10.2 *מודל בשלות התהליך*

הנדסת תוכנה היא דיסציפלינה צעירה מאד. ככזאת, ניטור רמות הבשלות הינו חלק מתמונת הנוף שלה לעוד שנים רבות.

מהן מגבלות הגישה המבוססת על מודלים?

הגישה המבוססת על מודלים מציגה קווי דמיון רבים לגישה המבוססת על תקנים. המודלים מכסים תחומים רחבים יותר מאשר תקני האיכות, ומשקפים ערכים עסקיים במידה נרחבת יותר. עם זאת, תהליך הערכת הביצועים מול מודל המוגדר חיצונית דומה בתמציתו. כתוצאה מכך, גם על גישה זו חלות אותן מגבלות בסיסיות.

הגישה המבוססת על מודלים היתה יכולה לטעון שהיא מחוללת פחות תקורות ביורוקרטיות ועלויות, אך ההערכה עדיין מהווה תכונה של גישה זו והיא התקורה הגבוהה מכולן. ההערכה מציבה גם בעיות אחרות. התחום הרחב של המודלים לניהול האיכות מצריך מעריכים בעלי ניסיון רחב מאוד, ובעיית הגיוס, ההדרכה ואורך תקופת שירות חריפה יותר אפילו מזו של גישה המבוססת על תקנים. הפרשנות הדקדקנית של המודל יכולה גם היא ליצור בעיות, ומבחינה פוטנציאלית אלו גרועות יותר מאלו של הגישה המבוססת על תקנים משום שהמודלים לניהול האיכות משפיעים על חלק גדול יותר של העסק.

ככלל, הגישה המבוססת על מודלים היתה צעד לכיוון הנכון, אלא שהיא לא התגברה על כל מגבלות הגישה המבוססת על תקנים.

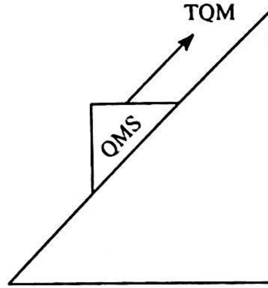
גישת שיפור תהליכים

גישת שיפור התהליך מושרשת היטב בתנועת **האיכות הכוללת** (Total quality - TQ). בפרק 6 דנו בנושאים העיקריים של הרעיונות והטכניקות, וגילינו שהמדידה היא נוהל המפתח של הגישה. שיפור מתמיד המבוסס על מדידיות הוא רעיון רב עוצמה, שיש בו פוטנציאל לא מוגבל לשיפורים, ועם זאת אינו מציג מודל של הנוהג הטוב ביותר.

תקן ISO 9000-3 מספק הנחיות לנוהג הקביל המזערי, ופעולה על פיו אינה יותר מאשר קו-בסיס מוצק לשיפורים. תקן ISO 9000-3 דן במדידת התהליך והמוצר לצורך תחזוקת מערכת ניהול האיכות, אך בפועל, אינו מגדיר דרישה לשיפור מתמיד.

כאשר קובעים את תקן ISO 9000-3 כקו-בסיס סביר עבור הנוהג שלנו, ומשתמשים בטכניקות של שיפורים מתמידים, יש לנו מנגנון לפיתוח מערכת איכות לטווח ארוך. תקן ISO 9000-3 ינחה אותנו לעבר מערכת איכות מתועדת, וזו תשמש לנו כמגן מפני היסחפות שיכולה לקרות בכל מערכת שאין בה כללים מוגדרים בבהירות. תרשים 10.3 מציג את תפקידה של מערכת איכות מתועדת בתור 'עוגן' שימנע את מערכת האיכות מפני השפעות הסחף. המערכת גוררת אחריה את העוגן כדי לוודא שכל רווח שיופק מהמנגנונים של השיפור המתמיד יהיה מוגן. משקל העוגן מייצג את התקורה הביורוקרטית של מערכת האיכות המתועדת. ללא ספק, עוגנים כבדים מאיטים את קצב ההתקדמות, אך גם שומרים על יציבות האוניה במים הסוערים.

תקן ISO 9000-3 אינו רק קלט שימושי עבור המערכת שלנו. יש תחום רחב של תקנים והנחיות של ISO, ANSI, ואחרים שניתן לגייסם לשירות בתור קו הבסיס שלנו כדי שישמשו כנוהג הטוב ביותר. מודלים של בשלות התהליך ומודלים של הערכת איכות מספקים שכבה נוספת של הנוהג המשופר, **והמבדקים** (benchmarking), או השוואת ביצועים, יכולים לספק מטרות אתגריות יותר בטווח הארוך.



תרשים 10.3 *מערכת ניהול האיכות (QMS) בתור עוגן*

תוכניות לשיפור מתמיד יכולות להתגבר על מגבלות הגישות המבוססות על תקנים וגישות המבוססות על מודלים, ועם זאת הן זקוקות לקו-בסיס של נוהג טוב. לתקנים ולמודלים יש אם כן תפקיד חיוני עדיין. לארגונים רבים, הנתיב המוביל יהיה כדלקמן:

- צור מערכת ניהול איכות מתועדת שעומדת בדרישות התקנים ISO 9001 ו- ISO 9000-3;
- צור תהליך לשיפור האיכות שיהיה מבוסס על מדידות;
- שפר את המערכת הבסיסית כדי שתהיה תואמת למודל נבחר מסוים, או שתגיע לרמת בשלות;
- המשך לשפר את המערכת לאחר מבדקים, תוך התקדמות לעבר מטרת הטווח הקצר שמיועדת להשיג את מתחריך המיידיים ולעבר מטרת הטווח הארוך, שהיא השגת אלה המוליכים בשוק.

גישה זו משתמשת בכל הגישות הזמינות כדי שהאחת תשלים את האחרת, וכדי ל מנגנון שיתגבר על רוב המגבלות של כל אחת מהגישות בפני עצמה.

לאן עכשיו?

לאן אנו הולכים עכשיו? כיצד נראה עתיד הנדסת התוכנה? לנוכח ההתקדמות הטכנית של העשור האחרון, קשה לחזות מה יכול לקרות בעשור הבא. הטכניקות והכלים ישתפרו ללא ספק, והשינוי עשוי להיות כה גדול עד שאי אפשר יהיה להכירם כשנשווה אותם לטכנולוגיה של ימינו.

דרושה מסגרת שתציג את הגישה ההתפתחותית (אבולוציונית) לפיתוח תוכנה, בכך שתיצור ארכיטקטורה ותאכלס אותה ביישומים הספציפיים המתאימים ככל שהם נעשים זמינים. בהקשר האיכות, הפעולה המקבילה תהיה יצירת מסגרת בה ניתן יהיה להכניס תקנים, מודלים, טכניקות, או כלים ככל שהם נעשים זמינים.

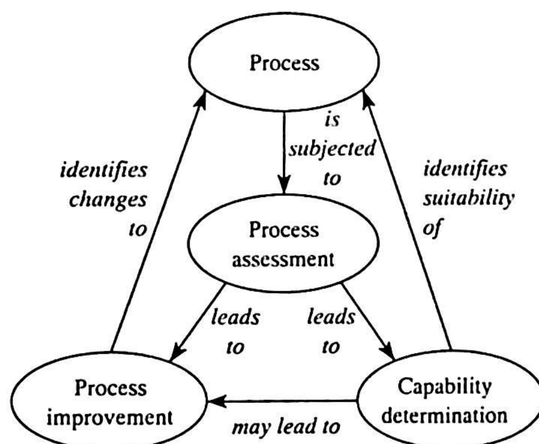
במקום להמשיך ולפתח כל הזמן קבוצה ספציפית של קווים מנחים כגון תקן ISO 9000-3, נוכל להגדיר קבוצה נוכחית של תקני קו-בסיס. אלה מהווים את ההגדרה הטובה ביותר של הנוהג הטוב, שתהיה זמינה לציבור הרחב באותה עת. הנוהג הטוב

ישתנה באופן בלתי נמנע, וצפוי שקצב השינויים יואץ במרוצת העשור הבא, אך המקור של הנוהג הטוב לא יהיה במסמך אחד יחיד. תהיה לנו גישה אל ספרייה שלמה של תקנים ומודלים ההולכים ומתפתחים ומשימת המפתח תהיה זיהוי של תת-הקבוצה של מסמכים אלה, שתהיה החשובה ביותר באותה עת.

גיבוש הגישות

בעת כתיבת שורות אלו מנסה הפרויקט - Software Process Improvement) SPICE Capability dEtermination), לכנס יחד את האלמנטים העיקריים של השיטות הקיימות. בתרשים 10.4 (וראה גם תרשים ח.5.1), משלבת גישת SPICE את תהליך ההערכה בידי מעריכים מאומנים, יחד עם הנחיות לנוהג הטוב ביותר, וקווים מנחים לשיפור התהליך. כיוון מקביל עוסק בנושא הגדרת היכולת, כך מתאפשרים זיהוי העוצמות והחולשות וקביעת מטרות ספציפיות לשיפורים.

פרויקט SPICE מגלם היבטים של גישות המבוססות על תקנים וגישות המבוססות על מודלים, והוא אמור לפחות לגבש מצב שבעבר התחרו בו למעשה זו בזו מספר גישות חלופיות. ברור פחות, אם יוכל לספק את היכולת ההתפתחותית המהירה שתאפשר לו לשקף דרך קבע את המצב הנוכחי של הטכנולוגיה והטכניקות ואת הנוהג הטוב ביותר בניהול האיכות.



***תרשים 10.4** מסגרת פרויקט SPICE*

מסגרת להתפתחות

מה צריכה להכיל **מסגרת התפתחותית** (evolutionary framework)? ודאי שעליה לשמר לפחות את כל החלקים הטובים שיש בגישות הקיימות. עם זאת, עליה להשתדל גם ליצור מבנה ארכיטקטוני בו יוכלו להיטמע בנקל כל הפיתוחים החדשים תוך מינימום מוחלט של תקורה מנהלית. ההשהיות הנאכפות היום על השיטות הנוכחיות עקב

הצורך בניסוח מסמכי התקנים ואישורים, תהיינה בלתי נסבלות בטכנולוגיה העתידית המואצת. מה תכיל הארכיטקטורה ההתפתחותית?

אין כל ספק שנצטרך למצוא דרכים טובות יותר לטיפול בפיתוח הלא-קווי האיטרטיבי, משום שברור שעוד נרבה לראות דרך פיתוח זו, אחריה יוסיפו לבוא שיטות עם מחזורי חיים אקזוטיים יותר.

שיפור תהליכים, מבוסס על מדידות, שיתבסס על תרבות **האיכות הכוללת** (TQ), יהיה ללא ספק הצעד הבא עבור אלה שאימצו כבר את מנגנון אישור ההסמכה. עבור אחרים שמפקפקים במנגנון זה, יהיה זה תחליף מתאים. עם זאת, כדי שהדבר יתאפשר, יהיה צורך לפתח נהגים לשיפור תהליכים שיתאימו לארגונים, משום שבשעה שהם מתחילים בתוכנית השיפורים, מודעותם לנושא האיכות מצומצמת מאוד, או כלל לא קיימת עדיין.

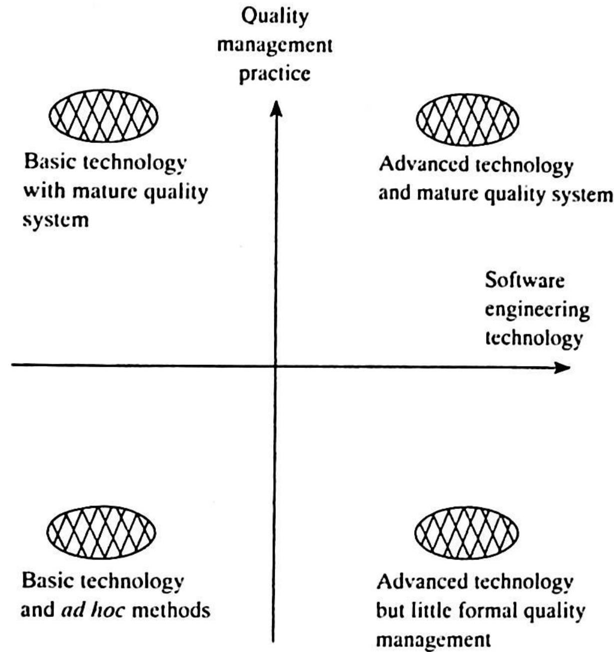
נוכל לדמיין לעצמנו עתיד בו יתפסו הארגונים מקומות שונים בטבלת האיכות. יהיו כאלה שינצלו את הטכנולוגיה המתקדמת ביותר, ואת נהגי ניהול האיכות המתקדמים באותה מידה; אחרים ישתמשו בטכנולוגיה הבסיסית ביותר, ולמעשה ללא נהגי איכות שיתמכו בהם; ויהיו גם כאלה שיימצאו בשלבים שונים שבין קטבים אלה. תרשים 10.5 מציג את הטבלה המציגה מיקומים שונים ברשת.

הניסיון מלמדנו שההתקדמות לעבר האזורים 'הטובים' של הטבלה רצוי שתיעשה בשלבים. כל ניסיון להתקדם רחוק מדי בכיוון מסוים, מבלי להתקדם במקביל גם לאורך הציר האחר מהווה הזמנה לכישלון. תופעה דומה נצפתה עם המעבר לשימוש בכלים לסיוע בשיפור התהליך. עודף אוטומציה בפרויקט שלא הוגדר כהלכה מוליך לסיכונים מוגברים; תמיכה מועטה מדי בשיפור התהליכים יכולה להוביל לעודף ביורוקרטיה.

כדי להיות שימושית בכל המצבים, חייבת מסגרת שיפור האיכות לאפשר לארגונים לזהות היכן הם נמצאים בשני הצירים, ולנסח תוכניות שיובילו אותם בכיוון הנכון. דבר זה משמעותו, לא רק הערכת יכולת התהליך אלא גם הערכת היכולת הטכנית ותמיכת הכלים, והגדרה ברורה של השאיפות העתידיות של הארגונים, במונחים של שיטות וטכניקות שלדעתם יצטרכו להיות מסוגלים לנצל.

כדי להיות אפקטיבית באמת, חייבת המסגרת להגדיר את הארכיטקטורה של עצמה ולדאוג לממשקים סטנדרטיים עבור כלים, טכניקות ומדידות שיהיו תואמים אותם. למשל, כל טכניקה חדשה לפיתוח תוכנה, יהיה צורך להגדיר במונחים של כלי הבקרה העיקריים שיוטבעו בה, המדידות העיקריות שתידרשנה לבקרה ולשפרה, והכלים שיימצאו מתאימים לתמיכה בה. בדומה לכך, כל כלי לשיפור תהליכים יהיה אמור להדגים את יכולתו לאסוף ולנתח את המדידות הקשורות לתהליכים הרלוונטיים.

המסגרת המתוארת כאן מרחיקה מאוד מעבר ליכולת הנוכחית, עם זאת היא מייצגת הצעה לפיתוח עתידי. זו השעה הנכונה לקבלת החלטות בדבר הכיוונים העתידיים. כשפרויקט SPICE יהיה זמין, הוא יספק תמיכה ביוזמות איכות קיימות, תוך תכנון ויישום יוזמות עתידיות.



תרשים 10.5 טבלת הבשלות האיכותית/טכנולוגית

בדומה לכל פיתוח התפתחותי, המסגרת מייצגת יותר משאפשר להשיג בפרויקט יחיד. היא מייצגת אסטרטגיה רחבה, ארוכת טווח שאפשר ליישמה ככל שהמשאבים נעשים זמינים. כל צעד בפיתוח יחשוף אפשרויות חדשות ובעיות שלא צפו קודם, ודבר זה יאפשר תכנון חוזר והתאמת היעדים הסופיים. הדבר החשוב מכל הוא נוכחותם המתמדת של יעדים לטווח ארוך המנחים את מאמץ הפיתוח של הטווח הקצר.

שאלות, שאלות

נחתום בהצעת מספר שאלות שיש לשקול בעת הגדרת אסטרטגיה לטווח ארוך. אלו שאלות שעלינו לשוב ולשאול תמיד, ושלעולם לא תהיה להן תשובה סופית.

- אילו שיטות הן הטובות ביותר?
- עד כמה חשובים התקנים?
- אילו כלים הם הטובים ביותר?
- איזה חלק מכל זה עלי לבצע?
- מה עלי למדוד?
- אילו טכניקות חדשות עלי ללמוד?

למעשה, אין סוף לשאלות, וכל תשובה תהיה זמנית בלבד. לכן חשוב לסקור כל הזמן את התגובותיך לשאלות אלו, כדי לוודא שחשיבתך מתקדמת יחד עם העולם סביבך.

על אף כל השינויים ואי-הוודאות, יש עדיין עקרונות כלליים שראוי לשקול, כדי לוודא שתוסיף להימשך כל העת אל כיוונים חדשים, כתגובה לאופנות המתחלפות.

העקרונות הנצחיים של פיתוח התוכנה

1. **המדידה כמעט שאינה מנוצלת עדיין**

מרבים לדבר היום אודות המדידה, אך עדיין קשה מאוד לגלות ברוב הארגונים פעילות ממשית בתחום זה. למי שיקדים ללמוד כיצד לנצל ביעילות את המדידה יהיה יתרון מוקדם וחשוב על פני מתחריו.

2. **השתמש בתקנים הקיימים בתור קו-בסיס**

בפרק 1 אמרנו שההנדסה היתה מבוססת על עקרונות רחבים ונצחיים. על אף כל החולשות שחשפנו בתקנים, בכל זאת הם מגלמים עקרונות נצחיים ממין זה ולכן, מספקים נקודת התחלה מוצקה מאוד בתחום האיכות.

3. **בחזרה אל היסודות**

העקרונות הנצחיים של ההנדסה (ושל מקצועות אחרים) אכן נצחיים, משום שהם מבטאים את הרעיונות הבסיסיים ביותר ללא כל תוספות. אין זה מקרה שהחזיקו מעמד במשך דורות. למד מהם. נסה לקבוע את קבוצת העקרונות שלך והערך את ארגונך מול עקרונות אלה.

4. **המשך לשוב ולהגדיר את האיכות**

ההגדרות צריכות להיעשות במונחים של הלקוח כדי שמאמצי האיכות שלך יצטרכו לעמוד כל הזמן באתגרים של רעיונות חדשים, אשר יאלצו אותך לשקול את השפעת מערכת האיכות שלך כלפי חוץ במקום כלפי פנים.

5. **המשך במאמצי ההערכה והתייצב מול הסיכונים הנוכחיים**

מה מאיים על איכות מוצריך? מהן הציפיות החדשות שלקוחותיך עתידים להביע בשבוע הבא או בשנה הבאה? מה עושים מתחריך כדי להעשיר את איכות מוצריהם?

6. **חפש כל הזמן אחר מבדקים (benchmarks)**

מדוד עצמך מול מבדקים אלה, כדי שתוכל לקיים את התמריץ לשיפורים.

7. **המשך לחפש דרך התפתחותית (אבולוציונית)**

המהפכות הן כמעט תמיד עקובות מדם וגם לא יציבות.

8. **הישמר מפני ידענים שמוכרים רק רעיונות**

אתה חייב למכור מוצרים.

9. **הישמר מפני מומחים להנדסת תוכנה שבא להם רעיון מבריק**

בשנה הבאה יהיה להם בוודאי רעיון חדש שיעשה את הרעיון הנוכחי למיושן.

10. **הישמר מפני פטנטים וגימיקים**

איכות אמיתית מוצגת כמעט תמיד בלשון ההמעטה - היא מדברת בעד עצמה.

נספח א׳

תקן ישראלי

ת״י 3-2000

ISO 9000-3

תקני ניהול איכות והבטחת איכות

הנחיות ליישום **ת״י 2001**

לשם **פיתוח, הספקה ותחזוקה של תוכנה**

הועתק ברשות מאת מכון התקנים הישראלי.
בכל מקרה מחייב התקן העדכני של מכון התקנים הישראלי.

כל הזכויות שמורות למכון התקנים הישראלי.
אין לצלם, להעתיק, לפרסם או לתרגם תקן זה או קטעים ממנו
ללא רשות מראש ובכתב ממכון התקנים הישראלי © 1992.

תוכן העניינים

מילות מפתח:

הבטחת איכות	quality assurance
מערכות הבטחת איכות	quality assurance systems
תכנות	programming
טכניקות תוכנה	software techniques
עיבוד נתונים	data processing

רשימת מונחים:

בדיקות תקדימיות	test cases
בדיקות קצה	boundary tests
גופן	font
כילולי	integrated
כילוליות	integration
תכן	design

0. הקדמה

עם התפתחות טכנולוגיית המידע גדלה כמות מוצרי התוכנה, וניהול האיכות של מוצרי התוכנה הפך חיוני. אחת הדרכים להקים מערכת ניהול איכות היא באמצעות הנחיות ברורות להבטחת איכות תוכנה.

הדרישות ממערכת איכות כללית, למצבים בהם שני צדדים חתומים על חוזה, פורסמו זה מכבר בתקן הישראלי ת״י 2001 - ״מערכות איכות - מודל להבטחת איכות בתיכון, בפיתות, בייצור, בהתקנה, במתן שירות ובתחזוקה״.

אולם תהליך הפיתוח והתחזוקה של תוכנה הינו שונה ממרבית הסוגים האחרים של המוצרים התעשייתיים. לכן, יש צורך לספק בתחום טכנולוגי זה, המתפתח במהירות, הנחיות נוספות למערכות איכות שמעורבים בהן מוצרי תוכנה. הנחיות אלו צריכות לקחת בחשבון את מצבה הנוכחי של טכנולוגיה זו.

אופי הפיתוח של תוכנה הוא כזה, שפעילויות מספר קשורות לשלבים מסוימים של תהליך הפיתוח, בעוד שאחרות מתבצעות בכל שלב של התהליך. מבנה ההנחיות נועד לשקף את ההבדלים הללו. אין התאמה ישירה בין תסדיר מסמך זה לתסדיר ת״י 2001, ולכן מובאים להלן שני אינדקסים (נספח א׳ ונספח ב׳), המיועדים לסייע בכל התייחסות לתקן זה.

חוזים בין שני צדדים למטרת פיתוח תוכנה יכולים להופיע בצורות שונות. במקרים מסוימים לא ניתן ליישם הנחיות אלו בחוזה בין שני צדדים, אפילו אם ״נתפרו״ במיוחד למטרה זו. לכן חשוב לקבוע אם ישנה התאמה מספקת בין יישומו של תקן זה לחוזה.

תקן זה דן בעיקר במצבים שבהם תוכנה ספציפית מפותחת כחלק מחוזה ובהתאם למפרט הלקוח. אולם התפישה המתוארת יכולה להיות בעלת ערך גם במצבים אחרים.

הערות:

1. שימוש בלשון זכר במסמך זה אינו אלא עניין של נוחות, וכל התייחסות לאנשים כוללת גם את הנשים. באופן דומה, שימוש ביחיד אינו בא להוציא מהכלל שימוש ברבים (ולהיפך), וזאת כאשר ההיגיון מרשה זאת.
2. בכל מקום במסמך זה שבו לא ניתנו הנחיות חדשות, הובא הטקסט של הסעיף הרלוונטי מתוך ת״י 2001 והוא מודפס בגופן (font) שונה.
3. במסמך זה יש מספר רשימות שאף לא אחת מהן אינה נחשבת מלאה. הרשימות מובאות כדוגמות בלבד.

1. תחום התקן

חלק זה של ת״י 2000 מציג הנחיות הבאות להקל על יישום ת״י 2001 בארגונים המפתחים תוכנה, מספקים תוכנה ומתחזקים אותה.

הכוונה היא לתת הנחייה במקרים שבהם חוזה בין שני צדדים מצריך הוכחה ליכולתו של הספק לפתח מוצרי תוכנה, לספקם ולתחזקם.

ההנחיות בחלק זה של ת"י 2000 מיועדות לתאר את הבקרות המוצעות ואת השיטות ליצירת תוכנה, שתעמודנה בדרישות הלקוח. הדבר מושג בעיקר על ידי מניעת אי התאמה בכל השלבים, החל מהפיתוח ועד לתחזוקה.

ההנחיות בחלק זה של ת"י 2000 מתאימים למצבים של חוזים למוצרי תוכנה שבהם:

א. החוזה דורש במפורש מאמץ תָּכֵן (design), והדרישות מהמוצר מוגדרות בעיקר במונחי ביצוע, או שיש להגדיר אותן.

ב. ביטחון במוצר יכול להיות מושג על ידי הדגמה נאותה של יכולת ספק מסוים לפתח מוצר תוכנה, לספקו ולתחזקו.

2. איזכורים

בתקנים המפורטים להלן ישנה הכנה לתוספות עתידיות. חלק מתוספות אלו מתמלא על ידי הפניות מטקסט זה. תוספות אלו מהוות שינוי למהדורה הקיימת של אותם תקנים ולמעשה הן מהוות מהדורות חדשות. בזמן הפרסום היו המהדורות המצוינות להלן בתוקף. כל התקנים צפויים לשינויים ועל הצדדים השותפים להסכמים המבוססים על תקנים אלה, לבדוק את האפשרות להשתמש במהדורות האחרונות של התקנים המפורטים מטה. לחברי IEC ו-ISO יש רשימה מעודכנת של התקנים הבין-לאומיים שבתוקף.

בסעיפים שונים של התקן הבין-לאומי ישנה הפניה לתקנים בין-לאומיים, במקום באים תקנים ישראליים כמפורט להלן:

ת"י 1432 (ISO 8402)	איכות - הגדרות מונחים.
ת"י 2000 (ISO 9000)	תקני ניהול איכות והבטחת איכות - הנחיות לבחירה ולשימוש.
ת"י 2001 (ISO 9001)	מערכות איכות - מודל להבטחת איכות בתיכון, בפיתוח, בייצור, בהתקנה, במתן שירות ובתחזוקה.
ת"י 2004 (ISO 9004)	מערכות איכות - אלמנטים במערכות איכות וניהול איכות: הנחיות.
ת"י 1080 חלק 1 (ISO 2382/1)	עיבוד נתונים אוטומטי - הגדרות מונחים: מונחים בסיסיים.

3. הגדרות

למטרות תקן בינלאומי זה יפה כוחן של הגדרות התקן הישראלי ת"י 1432 (ISO 8402), התקן הישראלי ת"י 1080 חלק 1 (ISO 2382/1) והגדרות אלו:

3.1 **תוכנה**: יצירה אינטלקטואלית אשר כוללת תוכניות, נהלים, חוקים וכל תיעוד הקשור אליהם, המתייחסים להפעלה של מערכת עיבוד נתונים.

הערה 4: התוכנה אינה תלויה במצע שעליו היא נרשמה.

3.2 **מוצר תוכנה**: מכלול תוכניות מחשב, נהלים, תיעוד נלווה ונתונים המיועדים למסירה למשתמש.

3.3 **פריט תוכנה**: חלק של מוצר תוכנה שאפשר לזהותו והנמצא בתהליך פיתוח או בסיומו.

3.4 **פיתוח**: כל הפעילויות הדרושות ליצירת מוצר תוכנה.

3.5 **שלב**: קטע מוגדר של עבודה.

הערה 5: שלב אינו מרמז על שימוש במודל של מחזור חיים כלשהו, ולא על פרק זמן בתהליך הפיתוח של מוצר תוכנה.

3.6 **אימות** (לתוכנה, verification): התהליך של הערכת המוצרים בשלב נתון, כדי להבטיח נכונות ועקיבות ביחס למוצרים ולתקנים שסופקו כקלט לשלב זה.

3.7 **הוכחת תקפות** (לתוכנה, validation): התהליך של הערכת תוכנה, הבא להבטיח התאמה לדרישות מפורטות.

4. מערכת איכות - מסגרת

4.1 אחריות ההנהלה

4.1.1 אחריות הנהלת הספק

4.1.1.1 מדיניות האיכות

הנהלת הספק תגדיר ותתעד את מדיניות האיכות שלה, את מטרותיה ואת מחויבותה לאיכות.

הספק יבטיח שמדיניות זו תובן, תיושם ותישמר בכל דרגי הארגון.

[ת"י 2001 : 1990, 4.1.1]

4.1.1.2 ארגון

4.1.1.2.1 אחריות וסמכות

חלוקת האחריות והסמכות ויחסי הגומלין שבין כל העובדים אשר מנהלים, מבצעים או מאמתים עבודות, המשפיעות על האיכות, יהיו מוגדרים היטב. חשוב להגדיר זאת במיוחד לגבי אותם עובדים, הזקוקים לעצמאות ארגונית ולסמכות לבצע את הפעולות שלהלן:

א. ליזום פעולות שימנעו אי-התאמה של המוצר.

ב. לזהות ולתעד בעיות באיכות המוצר.

ג. ליזום, להמליץ או לספק פתרונות בדרכים הקבועות בארגון.

ד. לוודא את יישום הפתרונות.

ה. לבקר את המשך העיבוד, ההספקה או ההתקנה של מוצר לא-מתאים עד שליקויו יתוקן, או עד שיתוקן מצב שהיה לא משביע רצון.

[ת"י 2001 : 1990, 4.1.2.1]

4.1.1.2.2 אימות - משאבים ועובדים

הספק יזהה את דרישות האימות התוך-ארגוניות ויקצה משאבים נאותים ועובדים מיומנים לפעילויות האימות (ראה סעיף 6.9). פעילויות האימות יכללו בחינה, בדיקה וניטור של תהליכי התיכון, הייצור, ההתקנה של המוצר או מתן השירות. סקרי תיכון ומיבדקים של מערכת האיכות, של התהליכים או של המוצר יבוצעו על ידי עובדים, שאינם תלויים באלה שלהם אחריות ישירה לעבודה המבוצעת.

[ת"י 2001 : 1990, 4.1.2.2]

4.1.1.2.3 נציג ההנהלה

הספק ימנה נציג הנהלה, אשר בלא קשר לתפקידיו האחרים, יהיה בעל סמכות ואחריות מוגדרות, כדי להבטיח שדרישות תקן זה ייושמו ויישמרו.

[ת"י 2001 : 1990, 4.1.2.3]

4.1.1.3 סקר ההנהלה

הנהלת הספק תסקור תקופתית את מערכת האיכות שנבחרה כדי להתאים לדרישות תקן זה, ותבטיח בכך את ההמשכיות של התאמת מערכת האיכות ואת האפקטיביות שלה. הרישומים של סקרים אלה יישמרו.

<u>הערה:</u> סקרי ההנהלה כוללים, בדרך כלל, הערכה של תוצאות מיבדקי האיכות הפנימיים, הנערכים על ידי הנהלת הספק או בשמה, להבדיל מביצוע מיבדק איכות פנימי על ידי העובדים והמנהלים האחראים ישירות למערכת האיכות.

[ת"י 2001 : 1990, 4.1.3]

4.1.2 אחריות הנהלת הלקוח

הלקוח ישתף פעולה עם הספק ויספק לו בזמן את כל המידע הדרוש. כמו כן יחליט הלקוח בנושאים העומדים על הפרק.

הלקוח ימנה נציג שיהיה אחראי לניהול משא ומתן עם הספק בנושאים הקשורים לחוזה. לנציג זה תהיה הסמכות, בהתאם לצרכים, לנהל משא ומתן עם הספק בנושאים הכוללים את הפרטים שלהלן, אך שאינם מוגבלים על ידם :

א. הגדרות הדרישות של הלקוח מהספק ;

ב. מתן תשובות לשאלות הספק ;

ג. אישור הצעות הספק;

ד. הבאה לידי הסכם עם הספק;

ה. הבטחה שארגונו של הלקוח מקיים את ההסכמים שנערכו עם הספק;

ו. הגדרת קריטריונים ונהלים לקבלה;

ז. טיפול ברכיבי תוכנה שסופקו ללקוח ונמצאו לא מתאימים לשימוש.

4.1.3 סקרים משותפים

יש לקבוע לוח זמנים לסקרים משותפים לספק וללקוח, כדי שיובטחו ההיבטים הבאים:

א. התאמת התוכנה למפרט הדרישות המוסכם עם הלקוח;

ב. תוצאות האימות;

ג. תוצאות בדיקת הקבלה.

התוצאות של סקרים מסוג זה יהיו מוסכמות ויתועדו.

4.2 מערכת איכות

4.2.1 כללי

הספק יקים ויקיים מערכת איכות מתועדת. מערכת האיכות תהיה תהליך כילולי (integrated) לכל אורך מחזור החיים, כדי להבטיח שהאיכות היא חלק מתהליך הפיתוח, ולא דבר המתגלה בסופו של התהליך בלבד. יש להתמקד במניעת בעיות, ולא בתיקון לאחר שהתגלתה בעיה.

הספק יבטיח שמערכת האיכות המתועדת תמומש בצורה יעילה.

4.2.2 תיעוד מערכת איכות

יש לתעד בצורה ברורה, שיטתית ומסודרת את כל אלמנטי מערכת האיכות, את הדרישות ואת ההוראות.

4.2.3 תוכנית איכות

הספק יכין תוכנית איכות ויתעדה, כדי לממש פעילויות איכות לכל פיתוח תוכנה על בסיס מערכת איכות, וכדי להבטיח שהיא מובנת ומקוימת על ידי הארגונים הנוגעים בדבר.

4.3 מיבדקי איכות פנימיים

הספק יפעיל מערך מקיף של מיבדקי איכות פנימיים, מתוכננים ומתועדים, כדי לאמת שפעילויות האיכות מתאימות לסידורים המתוכננים וכדי להעריך את האפקטיביות של מערכת האיכות.

מיבדקי האיכות יתוזמנו בהתאם למצבה ולחשיבותה של הפעילות.

עורכים את המיבדקים ואת פעולות המעקב לפי נהלים מתועדים.

מתעדים את תוצאות המיבדקים ומעבירים אותן לאחראים לתחומים שנבדקו. המנהל האחראי לתחום שנבדק ינקוט במועד פעולה לתיקון הליקויים שנתגלו במיבדק.

[ת"י 2001 : 1990, 4.17]

4.4 פעולה מתקנת

הספק יקים, יתעד ויקיים נהלים שישמשו כלהלן :

א. לחקור את הגורמים למוצר לא-מתאים ולפעולה המתקנת הנדרשת למניעת הישנותם.

ב. לנתח את התהליכים, שיטות העבודה, אישור החריגים, רשומות האיכות, דוחות השירות ותלונות הלקוחות, כדי לגלות ולסלק את הגורמים הפוטנציאליים למוצרים הלא-מתאימים.

ג. ליזום פעולות מנע לטיפול בבעיות ברמה המתאימה לסיכונים הצפויים.

ד. להפעיל בקרה כדי להבטיח שהפעולות המתקנות ננקטו ושהן אפקטיביות.

ה. ליישם ולתעד את השינויים בנהלים הנובעים מהפעולות המתקנות.

[ת"י 2001 : 1990, 4.14]

5. מערכת איכות - פעילויות במחזור החיים

5.1 כללי

פרויקט של פיתוח תוכנה יאורגן בהתאם למודל של מחזור החיים. פעילויות הקשורות לאיכות יתוכננו וימומשו בהתאם לאופי המודל של מחזור החיים שנבחר.

חלק זה של ת"י 2000 מיועד ליישום ללא קשר למודל של מחזור החיים שנבחר. אפשר לתת פירושים אחדים לכל תיאור, הנחיה, דרישה או מבנה, אך אין בכך כדי להצביע שהדרישה או ההנחייה מוגבלות למודל כלשהו.

5.2 סקר החוזה

5.2.1 כללי

הספק יקים ויקיים נהלים לסקר החוזה ולתיאום בין פעילויות אלו.

כל חוזה ייסקר על ידי הספק, כדי להבטיח מילוי תנאים אלה :

א. תחום החוזה והדרישות מוגדרים ומתועדים ;

ב. החלופות וסיכונים אפשריים מזוהים ;

ג. המידע שבבעלות הספק מוגן בצורה מספקת;

ד. כל דרישה השונה מזו הכלולה במכרז תידון בנפרד;

ה. לספק יש היכולת לעמוד בדרישות החוזה;

ו. אחריות הספק לעבודה של קבלני משנה מוגדרת;

ז. המינות מוסכם על שני הצדדים;

ח. ללקוח יש היכולת לעמוד במגבלות החוזיות.

רשומות של סקרי חוזה מסוג זה יישמרו.

5.2.2 סעיפי איכות בחוזה

בין השאר, בדרך כלל, מתאימים הסעיפים שלהלן להיכלל בחוזה:

א. קריטריונים לקבלה;

ב. טיפול בשינויים בדרישות הלקוח במהלך הפיתוח;

ג. טיפול בבעיות שהתגלו לאחר הקבלה, לרבות בטענות הקשורות לאיכות ובתלונות הלקוח;

ד. פעילויות המבוצעות על ידי הלקוח, ובמיוחד, תפקידו של הלקוח בעת הגדרת מפרט הדרישות, בעת ההתקנה ובעת הקבלה;

ה. אמצעים, כלים ופריטי תוכנה שיסופקו על ידי הלקוח;

ו. תקנים ונהלים שישתמשו בהם;

ז. מספר העותקים הנדרש (ראה 5.9).

5.3 מפרט דרישות הלקוח

5.3.1 כללי

כדי להתקדם בפיתוח התוכנה, יהיו בידי הספק דרישות פונקציונליות מפורטות וחד-משמעיות. נוסף על כך צריכים להיכלל בדרישות אלו כל ההיבטים הדרושים כדי לספק את צורכי הלקוח. להלן רשימה של היבטים אפשריים, אך שאינם מוגבלים לביצועים, בטיחות, אמינות, אבטחה ופרטיות. דרישות אלו יצוינו בצורה ברורה ומדויקת, כדי שניתן יהיה להוכיח תקיפות בעת קבלת המוצר.

דרישות אלו רשומות במפרט דרישות הלקוח. במספר מקרים מסופק מסמך זה על ידי הלקוח. אם לא, יפַתֵּח הספק דרישות אלו בשיתוף פעולה הדוק עם הלקוח. הספק יקבל את אישור הלקוח לפני היכנסו לשלב הפיתוח. מפרט דרישות הלקוח יהיה כפוף לבקרת תיעוד ולניהול תצורה כחלק מהתיעוד של שלב הפיתוח. במפרט יפורטו דרישות הלקוח במלואן, בצורה ישירה או על ידי הפניה, כל המישקים שבין מוצר התוכנה לבין תוכנות אחרות, או מוצרי חומרה.

5.3.2 שיתוף פעולה הדדי

במהלך פיתוח מפרט דרישות הלקוח, מומלץ לשים לב לנושאים אלה:

א. מינוי אנשים (משני הצדדים) האחראים לכתיבת מפרט דרישות הלקוח;

ב. שיטות הסכמה לדרישות ואישור שינויים;

ג. מאמצים למניעת אי הבנות באמצעים, כגון: הגדרת מונחים או הסברים הקשורים לרקע הדרישות;

ד. רישום תוצאות הדיונים על ידי שני הצדדים וסיקורן.

5.4 תכנון הפיתוח

5.4.1 כללי

תוכנית הפיתוח תכלול את הנושאים האלה:

א. הגדרת הפרויקט בהתייחס לפרויקטים קשורים של הלקוח, או של הספק, לרבות הצהרה על מטרות הפרויקט;

ב. ארגון משאבי הפרויקט, לרבות מבנה צוות הפיתוח, תחומי אחריות, שימוש בקבלני משנה ובמשאבים חומריים;

ג. שלבי הפיתוח, כמוגדר בסעיף 5.4.2.1;

ד. לוח זמנים של הפרויקט, אשר ניתן לזהות בו את המשימות לביצוע, את המשאבים והזמן הדרושים לכל משימה ולכל קשרי הגומלין בין המשימות;

ה. זיהוי תוכניות הקשורות בנושא, כגון:
- תוכנית איכות;
- תוכנית ניהול תצורה;
- תוכנית כילולית (integrated);
- תוכנית בדיקה.

תוכנית הפיתוח תעודכן ככל שהפיתוח מתקדם, וכל שלב יוגדר כמפורט בסעיף 5.4.2.1, לפני שמתחילים בפעילויות הקשורות לאותו שלב. השינויים יסוקרו ויאושרו לפני הביצוע.

5.4.2 תוכנית הפיתוח

5.4.2.1 שלבים

תוכנית הפיתוח תגדיר תהליך מסודר או שיטה להפיכת מפרט דרישות הלקוח למוצר תוכנה. לשם כך ייתכן שיהיה צורך לחלק את העבודה לשלבים ולזהות את הפרטים האלה:

א. שלבי פיתוח לביצוע;

ב. הקלטים הנדרשים בכל שלב;

ג. התוצרים הנדרשים מכל שלב;

ד. נוהלי אימות שיש להפעיל בכל שלב;

ה. ניתוח בעיות פוטנציאליות הקשורות לשלבי הפיתוח ולעמידה בדרישות המפורטות.

5.4.2.2 ניהול

תוכנית הפיתוח תגדיר כיצד ינוהל הפרויקט, לרבות זיהוים של פרטים אלה:

א. לוח הזמנים של הפיתוח, המימוש וההספקה;

ב. בקרת התקדמות העבודה;

ג. תחומי אחריות ארגוניים, הקצאת משאבים ומטלות עבודה;

ד. מישקים ארגוניים וטכניים בין קבוצות שונות.

5.4.2.3 שיטות וכלים לפיתוח

תוכנית הפיתוח תזהה שיטות שיבטיחו שכל הפעילויות יבוצעו בצורה נכונה. אפשר לכלול בתוכנית פרטים אלה:

א. כללים, נהגים ומוסכמות לפיתוח;

ב. כלים וטכניקות לפיתוח;

ג. ניהול תצורה.

5.4.3 בקרת התקדמות

סקרי התקדמות צריכים להיות מתוכננים, מבוצעים ומתועדים, כדי להבטיח שתפתרנה בעיות יוצאות דופן בהקצאת משאבים, וכדי להבטיח ביצוע אפקטיבי של תוכניות הפיתוח.

5.4.4 דרישות לשלבי הפיתוח

הדרישות לכל אחד משלבי הפיתוח יוגדרו ויתועדו. בעיות הקשורות בדרישות דו-משמעיות, סותרות או שאינן שלמות, ייפתרו עם האחראים לכתיבת הדרישות.

5.4.5 תוצר שלבי הפיתוח

התוצר הנדרש בכל שלב פיתוח יוגדר, יתועד ואומת ויעמוד בדרישות אלו:

א. ימלא אחר דרישות רלוונטיות;

ב. יכיל או יתייחס לקריטריוני קבלה, כדי להתקדם לשלבי הפיתוח העוקבים;

ג. יתאים לנהגים ולמוסכמות מתאימים של הפיתוח, בין אם פורטו במידע הקלט ובין אם לאו;

ד. יזהה את מאפייני המוצר החיוניים לבטיחותו ולפעולתו התקינה;

ה. יתאים לדרישות תחיקה ישימות.

5.4.6 אימות כל שלב

הספק יתווה תוכנית לאימות כל התוצרים של שלבי הפיתוח, בסוף כל שלב.

אימות הפיתוח יקבע שהתוצרים של שלבי הפיתוח יעמדו בדרישות המתאימות, באמצעים של בקרת פיתוח, כגון:

א. ביצוע סקרי פיתוח בנקודות מתאימות בשלבי הפיתוח;

ב. השוואה, אם אפשר, של תֶּכֶן חדש עם תכן דומה שהוכיח את עצמו;

ג. עריכת בדיקות והדגמות.

תוצאות האימות ופעולות נוספות שנדרשות כדי להבטיח שהדרישות המפורטות ממולאות, יירשמו וייבדקו כאשר מסתיימות הפעילויות הנוספות. רק תוצרי פיתוח מאומתים יימסרו לניהול תצורה ויתקבלו להמשך השימוש.

5.5 תכנון האיכות

5.5.1 כללי

כחלק מתוכנית הפיתוח יכין הספק תוכנית איכות.

תוכנית האיכות תעודכן תוך כדי התקדמות הפיתוח, והסעיפים הנוגעים לכל שלב יוגדרו במלואם לפני תחילת השלב.

תוכנית האיכות תיסקר פורמלית על ידי כל הארגונים המעורבים במימושה, והיא תהיה מוסכמת עליהם.

המסמך המתאר את תוכנית האיכות (ראה 5.5.2) יכול להיות מסמך עצמאי (שכותרתו 'תוכנית איכות'), או חלק ממסמך אחר, או יכול להיות מורכב מכמה מסמכים, לרבות תוכנית הפיתוח.

5.5.2 תוכן תוכנית איכות

תוכנית איכות תציין את הנושאים המפורטים להלן, או תתייחס אליהם:

א. יעדי האיכות, המבוטאים ככל האפשר, במונחים הניתנים למדידה;

ב. קריטריונים מוגדרים לדרישות ולתוצרים בכל שלב של הפיתוח;

ג. זיהוי סוגי פעילויות שיבוצעו לצורך בדיקה, אימות והוכחת תקפות;

ד. תכנון מפורט של פעילויות הבדיקה, האימות והוכחת התקפות שיבוצעו, לרבות לוח זמנים, משאבים וסמכויות מאשרות;

ה. תחומי אחריות ספציפיים לפעילויות איכות, כגון:
- סקרים ובדיקות;
- ניהול תצורה ובקרת שינויים;

- בקרת פגמים ופעולות מתקנות.

5.6 תֶּכֶן ומימוש

5.6.1 כללי

פעילויות התכן (design) והמימוש הן אלו המתמירות את מפרט הדרישות של הלקוח למוצר תוכנה. בגלל מורכבות מוצרי התוכנה חלה חובה לבצע פעילויות אלו בצורה ממושמעת, כדי ליצור מוצר בהתאם למפרט, ולא בהסתמך על פעילויות הבדיקה והוכחת התקפות להבטחת איכותו.

<u>הערה 6</u>: רמת החשיפה של המידע שיסופק ללקוח צריכה להיות מוסכמת הדדית על שני הצדדים, מכיון שבדרך כלל, תהליכי התכן והמימוש הינם סודות מסחריים של הספק.

5.6.2 תֶּכֶן

נוסף על הדרישות המשותפות לכל שלבי הפיתוח, יש להביא בחשבון את ההיבטים המפורטים להלן, שהם חלק בלתי נפרד מפעילויות התכן:

א. זיהוי שיקולי תכן: נוסף על מפרטי הדרישות והתוצרים ייבחנו היבטים, כמו כללי תכן והגדרות לממשקים פנימיים;

ב. שיטת תכן: יש להשתמש בשיטת תכן, המתאימה לסוג מוצר התוכנה שנמצא בפיתוח;

ג. שימוש בניסיון העבר בתחום התכן: יש להשתמש בלקחים שנלמדו מתוך ניסיון העבר בתהליכי תכן. הספק ימנע הישנות של בעיות דומות או זהות;

ד. התהליכים הבאים: תכן המוצר יאפשר בדיקה, תחזוקה ושימוש נוחים.

5.6.3 מימוש

נוסף על הדרישות המשותפות לכל פעילויות הפיתוח, יש להביא בחשבון את ההיבטים המפורטים להלן בכל פעילות מימוש:

א. כללים: יש לציין ולבחון כללים כמו כללי תכנות, שפות תכנות, שימוש עקבי בשמות, קידוד וכללי הוספת הערות;

ב. שיטת המימוש: הספק ישתמש בשיטות מימוש ובכלים מתאימים, כדי לעמוד בדרישות הלקוח.

5.6.4 סקרים

הספק יערוך סקרים כדי להבטיח שהמוצר עומד בדרישות ושהשיטות שהוזכרו לעיל מבוצעות בצורה נכונה. אין להמשיך בתכן, בביצוע, או בתהליך המימוש, עד אשר כל הליקויים הידועים יתוקנו בצורה משביעת רצון, או עד אשר יוודע הסיכון שבהמשך העבודה ללא תיקונים.

יש לשמור את הרשומות של סקרים כאלה.

5.7 בדיקה והוכחת תקיפות

5.7.1 כללי

ייתכן ויהיה צורך בבדיקות במספר רמות, מרמת פריט תוכנה בודד ועד לרמת מוצר תוכנה מוגמר. יש מספר גישות לבדיקה ולכילוליות (integration).

במספר מקרים ייכללו בפעילות אחת הוכחת תקפות (validation), ניסוי בתנאי שדה ובדיקות קבלה.

המסמך המתאר את תוכנית הבדיקה יכול להיות מסמך נפרד, חלק של מסמך אחר, או יכול להיות מורכב ממסמכים אחדים.

5.7.2 תכנון הבדיקה

הספק יקבע את תוכניות הבדיקה, המפרטים והנהלים ויסקור אותם לפני תחילת הבדיקות. יש להתחשב בפרטים אלה:

א. תוכניות פריט תוכנה, כילוליות, בדיקת מערכת ובדיקת קבלה;

ב. בדיקות מקדמיות (test cases), נתוני בדיקה ותוצאות צפויות;

ג. סוגי הבדיקות שיש לבצע. לדוגמה: בדיקות פונקציונליות, בדיקות קצה (boundary tests), בדיקות פעולה ובדיקות שימוש;

ד. סביבת בדיקה, כלים ותוכנת בדיקה;

ה. קריטריונים לבחינת סיום הבדיקות;

ו. תיעוד המשתמש;

ז. כות אדם הדרוש ודרישות הדרכה.

5.7.3 בדיקה

יש להקדיש תשומת לב מיוחדת להיבטי הבדיקה האלה:

א. תוצאות הבדיקה יירשמו כפי שמוגדר במפרט הרלוונטי;

ב. תצויין כל בעיה שהתגלתה והשפעתה האפשרית על חלקים אחרים של התוכנה. יש ליידע את האחראים לכך ולאפשר מעקב עד לפתרון;

ג. חלקים שהושפעו כתוצאה משינוי כלשהו יזוהו וייבדקו מחדש;

ד. יוערכו התאמת הבדיקה והרלוונטיות שלה;

ה. תצורות החומרה והתוכנה יישקלו ויתועדו.

5.7.4 הוכחת תקפות

לפני מסירת המוצר להפצה ולקבלת הלקוח, יוכיח הספק את תקפות פעולתו של המוצר כמוצר שלם וככל האפשר, בתנאים דומים לסביבת היישום כמצוין בחוזה.

5.7.5 ניסוי בתנאי שדה

כאשר יש צורך בניסוי בתנאי שדה, יש להתייחס לנקודות אלו:

א. התכונות שייבדקו בתנאי סביבה של השדה (field environment);

ב. תחומי האחריות הספציפיים של הספק והלקוח בנוגע לביצוע הניסוי ולהערכתו;

ג. החזרת סביבת המשתמש למצבה הקודם (לאחר הניסוי).

5.8 קבלה

5.8.1 כללי

כאשר הספק מוכן למסור את המוצר לאחר שעבר הוכחת תקיפות, ישפוט הלקוח אם אפשר לקבלו בהתאם לקריטריונים מוסכמים מראש ובאופן שצוין בחוזה.

שיטת הטיפול בבעיות המתגלות בעת הליך הקבלה וסילוקן תהיה מוסכמת על הלקוח ועל הספק ותתועד.

5.8.2 תכנון בדיקות קבלה

לפני ביצוע פעילויות קבלה, יעזור הספק ללקוח לזהות את הפרטים האלה:

א. לוח זמנים;

ב. נוהלי הערכה;

ג. סביבות תוכנה-חומרה ומשאבים;

ד. קריטריונים לקבלה.

5.9 שכפול, מסירה והתקנה

5.9.1 שכפול

שלב השכפול יבוצע לפני המסירה. לפני השכפול יש לתת תשומת לב לפרטים אלה:

א. מספר העותקים של כל פריט תוכנה שיש למסור;

ב. סוג המצע (מדיה) לכל פריט תוכנה, לרבות התסדיר והגירסה בצורה הניתנת לקריאה;

ג. קביעת התיעוד הדרוש, כגון ספרי עזר ומדריכים למשתמש;

ד. ענייני זכויות יוצרים ורישוי שנדונו וסוכמו;

ה. שמירת העותק המקורי ועותקי הגיבוי, אם אפשר, לרבות תוכנית להתאוששות ממצבי חירום;

ו. תקופת ההתחייבות של הספק לאספקת עותקים.

5.9.2 מסירה

יוכנו אמצעים שיאפשרו לאמת את נכונות העותקים של מוצר התוכנה שנמסרו ואת שלמותם.

5.9.3 התקנה

יש להגדיר בצורה ברורה את התפקידים, את תחומי האחריות ואת ההתחייבויות של הספק ושל הלקוח, ולהביא בחשבון את הפרטים האלה:

א. לוח זמנים, כולל שעות חריגות וסופי שבוע;

ב. אפשרות כניסה למתקני הלקוח (תגים, סיסמאות, ליווי);

ג. זמינות של עובדים מיומנים;

ד. זמינות של מערכות הלקוח וציודו וגישה אליהם;

ה. קביעה בחוזה את הצורך בהוכחת תקפות כחלק מההתקנה;

ו. נוהל פורמלי לאישור של כל התקנה לאחר סיומה.

5.10 תחזוקה

5.10.1 כללי

כאשר תחזוקה של מוצר תוכנה נדרשת על ידי הלקוח לאחר המסירה וההתקנה הראשונית, היא תעוגן בחוזה. הספק צריך ללהקים ולקיים נהלים לפעילויות תחזוקה ויאמת שהן תואמות לדרישות התחזוקה הספציפיות.

ממיינים את פעילויות התחזוקה למוצרי תוכנה, כלהלן:

א. פתרון בעיות;

ב. שינוי הממשק;

ג. הרחבת התפקוד, או שיפור ביצועים.

בחוזה יצוינו הפריטים שיש לתחזק ופרק הזמן שבו הם יהיו מתוחזקים. להלן דוגמאות של פריטים כאלה:

א. תוכנית או תוכניות;

ב. נתונים ומבניהם;

ג. מפרטים;

ד. תיעוד ללקוח או למשתמש;

ה. תיעוד לשימוש הספק.

5.10.2 תוכנית תחזוקה

כל פעילויות התחזוקה יבוצעו וינוהלו בהתאם לתוכנית תחזוקה מוגדרת ומוסכמת על הספק ועל הלקוח. התוכנית תכלול:

א. תחום התחזוקה;

ב. זיהוי המצב התחילי של המוצר;

ג. הארגון/ (או הארגונים) התומך;

ד. פעילויות תחזוקה;

ה. רישומי תחזוקה ודוחות.

5.10.3 זיהוי המצב התחילי של המוצר

המצב התחילי של המוצר המיועד לתחזוקה יהיה מוגדר, מתועד ומוסכם הן על ידי הספק והן על ידי הלקוח.

5.10.4 ארגון תומך

ייתכן שיהיה צורך להקים ארגון שיתמוך בפעילות התחזוקה ושיהיו בו נציגים הן של הספק והן של הלקוח. מכיון שהפעילויות בשלב התחזוקה אינן מתבצעות תמיד על בסיס של לוח זמנים קבוע, צריך ארגון זה להיות גמיש מספיק כדי להתמודד עם בעיות לא צפויות. ייתכן שיהיה גם צורך לזהות מתקנים ומשאבים שישמשו לפעילויות התחזוקה.

5.10.5 סוגים של פעילויות תחזוקה

כל שינויי תוכנה (מסיבות של פתרון בעיות, שינוי ממשק, הרחבת תפקוד ושיפור ביצועים) המבוצעים במהלך התחזוקה, ייעשו ככל האפשר בהתאם לאותם נהלים שישמשו לפיתוח מוצר התוכנה. כל השינויים האלה יתועדו בהתאם לנוהלי בקרת המסמכים וניהול תצורה.

א. פתרון בעיות: פתרון בעיות כרוך באיתור, ניתוח ותיקון של אי-התאמות תוכנה הגורמות לבעיות תפעול. אפשר לבצע תיקון זמני, כדי להקטין את זמן ההשבתה של המערכת, ולבצע שינויים קבועים בשלב מאוחר יותר.

ב. שינויי ממשק: שינויי ממשק עשויים להידרש, כאשר מבוצעים תוספות או שינויים במערכת החומרה או ברכיבים המבוקרים על ידי התוכנה.

ג. הרחבות פונקציונליות או שיפור ביצועים: הרחבות פונקציונליות או שיפור ביצועים עשויים להידרש על ידי הלקוח בשלב התחזוקה.

5.10.6 דוחות ורשומות תחזוקה

כל פעילויות התחזוקה יירשמו ויישמרו בתסדירים מוגדרים מראש. ייקבעו כללים מוסכמים להגשת דוחות תחזוקה.

רשומות התחזוקה יכללו את הפרטים שלהלן, לכל פריט תוכנה מתוחזק:

א. רשימת בקשות לסיוע, או דוחות תקלות שהתקבלו והמצב הנוכחי של כל אחד מהם;

ב. הארגון האחראי להיענות לבקשות סיוע או למימוש הפעילות המתקנות המתאימות;

ג. הקדימויות שנקבעו לפעולות המתקנות;

ד. תוצאות הפעולות המתקנות;

ה. נתונים סטטיסטיים על התרחשות תקלות ועל פעילויות תחזוקה.

רשומת פעילויות התחזוקה יכולה לשמש להערכת מוצר התוכנה ולשיפורו, ולשיפור מערכת האיכות עצמה.

5.10.7 נוהלי שחרור

הספק והלקוח יסכימו על נהלים להכנסת שינויים במוצר תוכנה, הנובעים מהצורך לשמור על רמת ביצועים, והם יתעדו נהלים אלה.

הנהלים יכללו:

א. כללי יסוד שיקבעו מתי יש צורך בשינויים (או תיקונים) זמניים ומתי יש צורך בעותק מלא ומעודכן של מוצר התוכנה;

ב. תיאור סוגי השחרורים (הגירסות) בהתאם לתדירות הופעתם ובהתאם להשפעה שיש להן על פעילות הלקוח ועל יכולתו לבצע שינויים בכל נקודת זמן כלשהי;

ג. שיטות שעל פיהן יונחה הלקוח בדבר שינויים נוכחיים או שינויים עתידיים;

ד. שיטות שיאפשרו לוודא שהשינויים שיבוצעו לא יגרמו לבעיות נוספות;

ה. דרישות לרשומות שמצוין בהן איזה שינויים מומשו ובאיזה מקומות, עבור אוסף של מוצרים ואתרים.

6. מערכת איכות - פעילויות תומכות

(ללא תלות בשלב כלשהו)

6.1 ניהול תצורה

6.1.1 כללי

ניהול תצורה מספק מנגנון לזיהוי, לבקרה ולמעקב אחר כל גירסה של כל פריט תוכנה.

במקרים רבים יש לתחזק ולבקר גם גירסות קודמות הנמצאות עדיין בשימוש.

המערכת לניהול תצורה צריכה:

א. לזהות בצורה ייחודית את הגירסות של כל פריט תוכנה;

ב. לזהות את הגירסות של כל פריט תוכנה, המרכיבות יחד גירסה ספציפית של מוצר שלם;

ג. לזהות את מצב הבניה של מוצרי תוכנה הנמצאים בפיתוח, או שסופקו והותקנו;

ד. לבקר עדכון של פריט תוכנה שנעשה סימולטנית על ידי יותר מאדם אחד;

ה. לתאם עדכון של אוסף מוצרים באתר אחד, או יותר, כנדרש;

ו. לזהות ולעקוב אחר כל הפעולות והשינויים הנובעים מבקשה לשינוי, החל מהייזום ועד לשחרור.

6.‏ תוכנית ניהול תצורה

פק יפתח ויממש ותוכנית ניהול תצורה, אשר תכלול:

א. גופים המעורבים בניהול תצורה והאחריות המוטלת על כל אחד מהם;

ב. פעילויות ניהול תצורה שיש לבצע;

ג. כלים, טכניקות ושיטות שיש להשתמש בהם בניהול תצורה;

ד. השלב שבו צריך להכניס פריטים לבקרת תצורה.

6.‏ פעילויות ניהול תצורה

6.1.3 זיהוי תצורה ומעקב

ספק יקבע ויקיים נהלים לזיהוי פריטי תוכנה במהלך כל השלבים, החל מהגדרת מפרטים דרך הפיתוח והשכפול ועד למסירה. אם נדרש בחוזה, אפשר להשתמש הלים אלה גם לאחר מסירת המוצר. לכל אחד מפריטי התוכנה יהיה זיהוי ייחודי.

להשתמש בנהלים כדי להבטיח שאפשר יהיה לזהות את הפרטים שלהלן עבור כל רסה של פריט תוכנה:

א. המפרטים הטכניים והפונקציונליים;

ב. כל כלי הפיתוח המשפיעים על המפרטים הטכניים והפונקציונליים;

ג. כל הממשקים לפריטי תוכנה אחרים ולחומרה;

ד. כל המסמכים וקובצי המחשב הקשורים לפריט התוכנה.

הוי פריט התוכנה יתנהל בצורה כזו, שניתן יהיה להראות את הזיקה שבין הפריט רישות החוזה.

בור מוצרים ששוחררו, יהיו נהלים שיקלו על המעקב אחר פריט התוכנה או אחר מוצר.

6.1.3 בקרת שינויים

הספק יקבע ויקיים נהלים לזיהוי, לתיעוד, לסיקור ולהרשאה של כל השינויים בפריטי התוכנה הנמצאים בניהול תצורה. כל השינויים בפריטי התוכנה יבוצעו בהתאם לנהלים אלה.

לפני שמתקבל שינוי, תאושר תקפותו, תזוהה השפעתו על פריטים אחרים ותיבדק.

יסופקו שיטות כדי לידע את הנוגעים בדבר לגבי השינויים ולהראות להם את העקיבות בין השינויים לבין החלקים של פריטי התוכנה ששונו.

6.1.3.3 דוח מצב תצורה

הספק יקבע ויקיים נהלים לרישום, לניהול ולדיווח של מצב פריטי התוכנה, של הבקשות לשינוי ושל מימוש השינויים המאושרים.

6.2 בקרת מסמכים

6.2.1 כללי

הספק יקבע ויקיים נהלים לבקרת כל המסמכים המתייחסים לתוכן חלק זה של ת"י 2000. אלה כוללים:

א. הגדרת המסמכים שנוהלי בקרת מסמכים יחולו עליהם;

ב. אישור נהלים והוצאתם;

ג. נוהלי שינוי הכוללים דחייה ולפי הצורך שחרור.

6.2.2 סוגי מסמכים

הנהלים לבקרת מסמכים יחולו על המסמכים הרלוונטיים, כלהלן:

א. מסמכי נוהל המתארים את מערכת האיכות, שיש להחיל במחזור החיים של התוכנה;

ב. מסמכי תכנון המתארים את התכנון של כל פעילויות הספק ואת התקדמותן, ואת פעולות הגומלין שלו עם הלקוח;

ג. מסמכי המוצר המתארים מוצר תוכנה מסוים, לרבות:
- דרישות לשלבי הפיתוח;
- תוצרים של שלבי הפיתוח;
- תוכניות אימות ותקפות ותוצאותיהן;
- תיעוד ללקוח ולמשתמש;
- תיעוד תחזוקה.

6.2.3 אישור מסמכים והוצאתם

כל המסמכים צריכים להיסקר ולעבור אישור של צוות עובדים מורשה לפני הוצאתם. יש להגדיר נהלים שיבטיחו:

א. המסמכים המתאימים יהיו זמינים במקומות שבהם מבוצעות פעולות החיוניות לתפקוד האפקטיבי של מערכת האיכות ;

ב. מסמכים שפג תוקפם יסולקו מיד מכל האתרים שבהם נופקו או יושמו.

כאשר משתמשים בקובצי מחשב, יש להקדיש תשומת לב מיוחדת לנוהלי אישור, גישה, הפצה ושמירה בארכיון.

6.2.4 שינויים ועדכונים במסמכים

שינויים במסמכים ייסקרו ויאושרו על ידי אותם בעלי תפקיד, יחידות וארגונים מוסמכים, שסקרו את המהדורה המקורית ואישרו אותה, אלא אם מונו אחרים לבקר את השינויים והעדכונים. לארגונים המוסמכים תהיה גישה למידע רקע ישים, שעליו יבססו את סיקורם ואישורם.

במקום שהדבר מעשי, יזוהה אופי השינוי במסמך או בנספחים נאותים.

רשומות-אב או מסמך שקיל לבקרת תהליך התיעוד יוכנו, כדי לזהות את הגירסה המעודכנת של מסמכים וכדי למנוע שימוש במסמכים שְאינם ישימים.

מסמכים ייערכו מחדש ותפורסם מהדורה חדשה שלהם, לאחר עריכת מספר סביר של שינויים ועדכונים.

[ת"י 2001 : 1990, 4.5.2]

6.3 רשומות האיכות

הספק יקים ויקיים נהלים לזיהוי, לאיסוף, למספור, לתיוק, להחסנה, לתחזוקה ולהמשך הטיפול ברשומות האיכות.

הספק ישמור את רשומות האיכות ויציג אותן כראיות להשגת האיכות הנדרשת ולהפעלה האפקטיבית של מערכת האיכות. רשומות איכות של קבלני המשנה הנוגעות למוצר יהוו חלק מהרשומות שהוזכרו לעיל.

כל רשומות האיכות יהיו ברורות וקריאות וניתנות לזיהוי כלפי המוצר.

מחסינים את רשומות האיכות, שיהיו נגישות בקלות. מחסינים אותן בסביבה מוגנת, המקטינה את קלקולן, את הפגיעה בהן ואת אובדנן. זמן השמירה של רשומות האיכות ייקבע ויתועד. כאשר הדבר הוסכם בחוזה, תהיה תקופה מוגדרת, שבה יהיו רשומות האיכות נגישות לבדיקה על ידי הלקוח או על ידי נציגו.

[ת"י 2001 : 1990, 4.16]

6.4 מדידה

6.4.1 מדידת מוצר

מדדים ידווחו וישמשו לניהול תהליך הפיתוח והמסירה, ויהיו רלוונטיים למוצר התוכנה המסוים.

אין כרגע דרכי מדידה לאיכות תוכנה, המקובלים באופן אוניברסלי. אולם כמינימום יש להשתמש במדדים כלשהם המייצגים דיווח של מספר כשלים בשדה, או את מספר הפגמים מנקודת מבטו של הלקוח. המדדים הנבחרים יתוארו באופן שאפשר יהיה להשוות את התוצאות.

הספק של מוצרי תוכנה יאסוף נתוני מדידה כמותיים על איכות מוצרי התוכנה. יינקטו אמצעים אלה כדי:

א. לאסוף נתונים ולדווח באופן קבוע על ערכי המדדים;

ב. לזהות את רמת הביצועים הנוכחית בכל מדד;

ג. לבצע פעולה מתקנת, אם רמות המדד יורדות אל מעבר לרמה שהוגדרה מראש, או עולות עליה;

ד. לקבוע יעדי שיפור מוגדרים במונחים של המדדים.

6.4.2 מדידת תהליך

לספק יהיו מדדים כמותיים של איכות תהליכי הפיתוח והמסירה. מדדים אלה ישקפו נתונים אלה:

א. באיזו מידה התנהל תהליך הפיתוח לפי אבני דרך שנקבעו, ובאיזו מידה הושגו יעדי איכות במהלך התהליך בהתאם ללוח הזמנים;

ב. מידת היעילות של תהליך הפיתוח בהקטנת ההסתברות להיווצרות תקלות שיתגלו, או שלא יתגלו.

כאן, בדומה למדדי המוצר, הדבר החשוב הוא שרמות המדדים ידועות ומשמשות לבקרת התהליך ולשיפורו, ולא באיזה מדדים משתמשים. המדדים הנבחרים יתאימו לתהליך ואם אפשר, ישפיעו ישירות על איכות התוכנה שיש לספק. ייתכן ומדדים שונים יתאימו למוצרי תוכנה שונים המיוצרים על ידי אותו ספק.

6.5 כללים, נהגים ומוסכמות

הספק יספק כללים, נהגים ומוסכמות כדי להפוך את מערכת האיכות המתוארת בחלק זה של ת״י 2000, לאפקטיבית. הספק יבדוק את הכללים והמוסכמות ויעדכנם הצורך.

6.6 כלים וטכניקות

הספק ישתמש בכלים, אמצעים וטכניקות, כדי להפוך את הנחיות מערכת האיכות המתוארת בחלק זה של ת״י 2000, לאפקטיבית. כלים, אמצעים וטכניקות אלה יכולים להיות אפקטיביים הן לצורכי ההנהלה והן לפיתוח המוצר. הספק ישפר כלים וטכניקות אלה לפי הצורך.

6.7 רכש

6.7.1 כללי

הספק יבטיח שהמוצר או השירות הנרכשים עומדים בדרישות המצוינות.

מסמכי הרכש יכילו מידע המתאר בצורה ברורה את המוצר ואת השירותים שהוזמנו. לפני שחרור המוצר יסקור הספק את מסמכי הרכש ויבדוק את התאמתם לדרישות הספציפיות, ויאשרם.

הערה 7: מוצר נרכש יכול להיות פריט חומרהאו תוכנה, המיועד להיכלל במוצר הסופי, או כלי המיועד לעזור בפיתוח המוצר הנדרש.

6.7.2 הערכה של קבלני משנה

הספק יבחר לו קבלני משנה לפי יכולתם להתאים לדרישות קבלנות המשנה ודרישות האיכות בכלל זה. הספק יקים ויקיים רשומות של קבלני משנה קבילים.

בחירת קבלני המשנה, אופי הבקרה והיקפה, כפי שתופעל על ידי הספק, יהיו תלויים בסוג המוצר ולפי הצורך ברשומות על הביצועים והיכולת המוכחים של קבלני המשנה בעבר.

הספק יבטיח שהבקרה של מערכת האיכות אפקטיבית.

[ת"י 2001 : 1990, 4.6.2]

6.7.3 הוכחת תקפות המוצר הנרכש

הספק אחראי להוכחת תקפות עבודתו של קבלן המשנה. ייתכן ולשם כך יידרש קבלן המשנה לתאם את התכן וסקרים אחרים עם מערכת האיכות של הספק. אם כך, ייכללו דרישות אלו בחוזה המשנה. כן תיכלל בחוזה המשנה כל דרישה לבדיקות קבלה שיערוך הספק על תוצאות עבודת קבלני המשנה.

כאשר מצוין בחוזה, יש לאפשר ללקוח או לנציגו לבחון במהלך העבודה אצל קבלן המשנה או בעת קבלת המוצר, אם המוצר עומד בדרישות המצוינות. הוכחת תקיפות על ידי הלקוח, אינה משחררת את הספק מאחריות לספק מוצר ראוי לשימוש, ואינה מוציאה מכלל אפשרות דחיית המוצר בעתיד.

כאשר הלקוח או נציגו בוחרים להוכיח תקפות באתר של קבלן המשנה, אין הספק יכול להשתמש בכך כעדות לבקרה אפקטיבית של האיכות הנעשית על ידי קבלן המשנה.

6.8 מוצר תוכנה משולב

ייתכן שהספק יידרש להשתמש במוצר תוכנה שסופק על ידי הלקוח, או על ידי צד שלישי. הספק יקבע ויקיים נהלים להוכחת תקפות, אחסון, הגנה ותחזוקה של מוצר כזה. יש לתת את הדעת לתמיכה במוצר תוכנה כזה בכל חוזה תחזוקה המתייחס למוצר הסופי.

מוצר שסופק על ידי הלקוח ונמצא בלתי מתאים לשימוש יירשם וידווח על כך ללקוח. הוכחת תקפות על ידי הלקוח אינה משחררת אותו מהאחריות לספק מוצר ראוי לשימוש.

6.9 הדרכה

הספק יקבע ויקיים נהלים לזיהוי צורכי ההדרכה ויאפשר הדרכה לכל העובדים העוסקים בפעילויות המשפיעות על האיכות. המועסקים במשימות מסוימות יוסמכו על בסיס השכלה נאותה, הכשרה, או לפי הניסיון המצטבר שרכשו.

הנושאים שיש להתייחס אליהם יוגדרו בהתאם לכלים, לטכניקות ולשיטות ספציפיים ובהתאם למשאבי המחשב שישתמשו בהם לפיתוח ולניהול של מוצר תוכנה ולניהולו.

ייתכן שיהיה צורך לכלול הדרכה, הכוללת הקניית מיומנויות וידע הקשורים לתחום המיוחד שהתוכנה תטפל בו.

יש לשמור על רישומים הנוגעים להדרכה ולניסיון.

<u>נספח א</u> (לידיעה בלבד)

הפניה הדדית בין סעיפי ת״י 3-2000 לבין ת״י 2001.

הסעיף בת״י 3-2000		הסעיף בת״י 2001
4.1	אחריות ההנהלה	4.1
4.2	מערכת איכות	4.2
4.3	מבדקי איכות פנימיים	4.17
4.4	פעולה מתקנת	4.14
5.2	סקר החוזה	4.3
5.3	מפרט דרישות הלקוח	4.3, 4.4
5.4	תכנון הפיתוח	4.4
5.5	תכנון האיכות	4.2, 4.4
5.6	תכן ויישום	4.4, 4.9, 4.13
5.7	בדיקה והוכחת תקפות	4.4, 4.10, 4.11, 4.13
5.8	קבלה	4.10, 4.15
5.9	שכפול, מסירה והתקנה	4.10, 4.13, 4.15
5.10	תחזוקה	4.13, 4.19
6.1	ניהול תצורה	4.4, 4.5, 4.8, 4.12, 4.13
6.2	בקרת מסמכים	4.5
6.3	רשומות איכות	4.16
6.4	מדידה	4.20
6.5	כללים, נוהגים ומוסכמות	4.9, 4.11
6.6	כלים וטכניקות	4.9, 4.11
6.7	רכש	4.6
6.8	מוצר תוכנה משולב	4.7
6.9	הדרכה	4.18

נספח ב (לידיעה בלבד)

הפניה הדדית בין סעיפי ת״י 2001 לבין ת״י 3-2000.

הסעיף בת״י 2001		הסעיף בת״י 3-2000
4	דרישות מערכת איכות	4, 5, 6
4.1	אחריות ההנהלה	4.1
4.2	מערכת איכות	4.2
4.3	סקר החוזה	5.2, 5.3
4.4	בקרת התכן	5.3, 5.4, 5.5, 5.6, 5.7, 6.1
4.5	בקרת התיעוד	6.1, 6.2
4.6	רכש	6.7
4.7	מוצר המסופק על ידי הלקוח	6.8
4.8	זיהוי המוצר ועקיבה	6.1
4.9	בקרת תהליך	5.6, 6.5, 6.6
4.10	בחינה ובדיקה	5.7, 5.8, 5.9
4.11	ציוד הבחינה, המדידה והבדיקה	5.7, 6.5, 6.6
4.12	מצב הבחינה והבדיקה	6.1
4.13	בקרה של מוצר שאינו תואם	5.6, 5.7, 5.9, 6.1
4.14	פעולה מתקנת	4.4
4.15	שינוע, החסנה, אריזה והספקה	5.8, 5.9
4.16	רשומות האיכות	6.3
4.17	מבדקי איכות פנימיים	4.3
4.18	הדרכה	6.9
4.19	שירות	5.10
4.20	טכניקות סטטיסטיות	6.4



התאחדות התעשיינים
בישראל



משרד ראש הממשלה
המרכז לאיכות ומצוינות

משרד התעשיה והמסחר



איגוד תעשיות
האלקטרוניקה

נספח ב׳

הפרס הלאומי לאיכות בתעשיה

תקנון

מבוא

הצטיינות היא תנאי להצלחה עיסקית ואף לשרידות בשנות ה- 90. האיכות הכוללת של חברה תעשייתית מקיפה את מוצריה, שירותיה, כל מרכיבי תפקודה הפנימי והחיצוני. הצטיינות תושג כתוצאה מניהול לאיכות על בסיס תהליכי שיפור מתמיד בכל רמות ופעילויות הארגון.

מתוך הכרה בעובדות אלה החליטה ממשלת ישראל כי ראש הממשלה יעניק מדי שנה את פרס האיכות בתעשיה למפעלים מצטיינים.

את התקנון המפורט לקבלת הפרס אפשר לקבל בוועדת פרס האיכות / איגוד תעשיות האלקטרוניקה, ת.ד. 50026 תל-אביב 61500. בטלפון : 03-5198862/3

מטרות הענקת הפרס

- לתת פומבי ולפתח את המודעות לחשיבותה של איכות כוללת.
- דירבון חברות וארגונים להיכנס לתהליך קבוע של שיפור.
- עידוד שיתוף פעולה בין חברות כמנוף לשיפור תחרותיות בעולם.
- לפרסם בארץ ובעולם שהתעשיה בישראל שואפת ופועלת להשגת הצטיינות.
- להתוות את הדרך להשגת איכות באמצעות הקריטריונים של הפרס.

- להקנות לזוכים כלי שיווקי בארץ ובעולם ולהוות הכרה בהישגיהם.

אמות מידה ומדדים

המאפיינים של ארגונים מצטיינים והקריטריונים על פיהם תבוצע הערכת מועמדים הם:

- מנהיגות לאיכות הכוללת מעורבות מנהלים, קביעת מדיניות איכות, תכנון אסטרטגי לאיכות ותרומה לסביבה.
- שביעות רצון של הלקוחות.
- קיום תהליך מתמשך של שיפור.
- תוצאות איכות בפועל.
- תשתית לאיכות.
- שיפור ומיצוי המשאב האנושי.
- איכות במידע ומערכות דוות.

הקריטריונים יבדקו ויעודכנו במידת הצורך מדי שנה.

השיטה

א. התחרות פתוחה לחברות ומפעלים בכל גודל וליחידות משנה של חברות בתנאי שאלה מקיפות לפחות תחום עסקים מסויים ומהוות יחידות עיסקיות (Business Units), דהיינו כוללות את מגוון הפעילויות המאפיינות עסק העומד בזכות עצמו.

התחרות תתקיים בשתי הקבצות:

◊ מפעלים קטנים - עד 150 עובדים במפעל.

◊ מפעלים גדולים יותר.

ב. הזוכים בפרס יעזרו במידע והדרכה למפעלים אחרים ליישם מערכות איכות כוללת.

טופס הערכה - קריטריונים לבחינת זכאות לפרס

(סה"כ 1000 נקודות)

הערה: הניקוד משמש מדד חשיבות יחסי לנושא ההערכה.

1. מנהיגות לאיכות (150 נקודות)

1.1 מדיניות איכות ברורה ומחייבת ליצירת תרבות איכות

1.1.1 מדיניות האיכות והדרך בה מדיניות זאת מועברת ומושרשת לכלל עובדי הארגון.

1.1.2 מעקב אחרי חדירת הערכים לכל רובדי הארגון.

1.2 יעדי איכות הנגזרים מן המדיניות

1.2.1 שילוב יעדי איכות הנגזרים ממדיניות האיכות, בתכנון לטווח קצר וארוך.

1.2.2 מעקב אחר שילוב יעדי איכות אלה בכל רובדי הארגון וההתקדמות להשגתם.

1.3 מחויבות לאיכות ומעורבות אישית של ההנהלה הבכירה

1.3.1 פעולתה של ההנהלה הבכירה ופעולתם של חברי ההנהלה בהשרשת תרבות האיכות החל מתכנון והצבת יעדים, דרך מעקב ביצוע, וסייס בהכרה בהישגי איכות.

1.3.2 השתתפות אישית של מנהלים בכירים בצוותי שיפור פנימיים ובקידום האיכות במסגרות הרחבות.

1.4 הכרת מדיניות ויעדי האיכות על ידי כל רובדי הארגון

1.4.1 מסגרות בהן מועברים ונדונים מדיניות ויעדי האיכות של הארגון.

1.4.2 מעקב אחר הכרת המדיניות והיעדים על ידי כל רובדי הארגון.

1.4.3 השתלבות ערכי האיכות והיעדים בפעילות היום-יומית.

1.5 ניהול לאיכות

1.5.1 השרשת נושא שילוב האיכות בכל רמות הניהול.

1.5.2 שיתוף פעולה ועבודת צוות בכל דרגי הניהול לקיום ושיפור האיכות.

1.5.3 מעקב, מדידה ופעולות תיקון ליישום העקרונות הנזכרים.

1.6 שילוב איכות בתכנון לטווח קצר

1.6.1 קביעת יעדי הארגון להבטחת תחרותיות והצטיינות.

1.6.2 תהליכי איסוף ועיבוד מידע לקביעת היעדים האלה.

1.6.3 קיום ושיפור תהליך מוגדר של תכנון לטווח ארוך עם דגש על איכות.

1.7 סיוע לספקים לעמוד בדרישות האיכות

1.7.1 בחירת ספקים על בסיס איכות, ולא רק מחיר.

1.7.2 טיפוח הקשרים עם הספקים ושילובם בתוכניות האיכות הכוללת של הארגון.

1.7.3 הגדרת מדד לאיכות ספקים והנחיית הספקים בשיפורו.

1.7.4 תהליך צמצום בדיקות קבלה על ידי הבטחת איכות המוצרים והשרותים המסופקים בידי הספקים.

1.8 תמיכה והשתתפות בגופי תקינה, צוותי חשיבה, חינוך והוראה לאיכות

1.8.1 השתתפות עובדי הארגון בגופי תקינה, חינוך והדרכה, תוך שימת דגש על נושא האיכות.

1.8.2 טיפוח תודעת האיכות והחלפת מידע עם גופים חיצוניים ומוסדות ציבור בסביבה הקרובה והרחוקה.

1.9 הקפדה על שיפור האיכות

1.9.1 תרומת הארגון לנושאי בריאות, בטיחות ואיכות הסביבה, הן בין עובדי המפעל והן מחוצה לו.

.2 שביעות רצון לקוחות, פנימיים וחיצוניים (200 נקודות)

2.1 קיום מערכת לאיפיון צורכי הלקוחות וציפיותיהם

2.1.1 זיהוי דרישות וציפיות הלקוחות לטווח קצר וארוך.

2.1.2 הגדרת מגזרי שוק, איפיון הצרכים והגדרת המוצרים .

2.1.3 מעקב אחר התאמת התחזיות לתוצאות .

2.2 קיום מערכת למדידת שביעות רצון הלקוח

2.2.1 שיטות ומדדים המשמשים לאבחון שביעות רצון הלקוחות.

2.2.2 השימוש באלה להפקת לקחים והגדרת פעולות תיקון.

2.2.3 הדרכים והשיטות בהם מעביר הארגון ללקוחותיו את חשיבותם וחשיבות דעתם בעיני הארגון.

2.3 שימוש אפקטיבי ויעיל בממצאי שביעות רצון הלקוחות

2.3.1 שיטות ומדדים לריכוז תלונות הלקוחות והשימוש בהם למטרות הגדרת פעולות תיקון ושיפור המוצרים/השירות.

2.3.2 מודעות כלל עובדי הארגון לחשיבות שביעות רצון הלקוחות ולמדדים הרלוונטיים.

2.4 מתן משוב ללקוחות (פנימיים וחיצוניים)

2.4.1 שיטה וזמן תגובה ומעקב למתן משוב, הערות לבקשות, תלונות ודרישות לקוחות.

2.4.2 רמת המודעות של ההנהלה וכלל העובדים לנושא.

2.5 שיפור בפועל של שביעות רצון לאורך זמן

2.5.1 דוגמאות למגמות במדדים של שביעות רצון לקוחות שהוזכרו בסעיפים קודמים.

3. קיום תהליך מתמשך של שיפור (150 נקודות)

3.1 יעדים מוגדרים לשיפור בכל תחומי פעילות הארגון

3.1.1 יעדי השיפור שהוגדרו, סיבת בחירתם והדרך בה הוגדרו (שיתוף עובדים, החלטת הנהלה, אילוצים חיצוניים).

3.2 מערכת מדידת נתונים ודיווח של ממצאים ומגמות

3.2.1 בסיסי הנתונים התומכים במערכת השיפורים האלה.

3.3 שיטות ותהליכים לשיפור

3.3.1 תהליכי השיפור (ישיבות ו/או דוחות תקופתיים, חד-פעמיים, צוותי חשיבה וכדי׳).

3.3.2 הדרך בה מובטחת השרשת תהליכים אלה בתרבות המפעל.

3.4 מעורבות מנהלים ועובדים בתהליכי השיפור

3.4.1 השתתפות של מנהלים בכירים והערכתם.

3.4.2 יצירת אווירה המעודדת עבודת צוות.

3.5 תוצאות בפועל של פעולות שיפור

3.5.1 דוגמאות למגמות ותוצאות בפועל של פעולות שיפור.

4. תוצאות איכות בפועל (160 נקודות)

4.1 דיווחי איכות כמותיים וקבועים, ניתוחם הסטטיסטי ונקיטת פעולות תיקון ומֶנַע

4.1.1 השיטה/הדרך בה נבחרים הפרמטרים ומקורות הדיווח.

4.1.2 שיטות העיבוד ומעקב אחר פעולות מנע/תיקון עליהן הוחלט.

4.2 שילוב נושאי האיכות בנושאים תפעוליים

4.2.1 שילוב נושאי האיכות (בצורה כמותית) בנושאים תפעוליים בכל תחומי הפעילות.

4.3 רמת הכיסוי והפירוט של מדידת האיכות

4.3.1 פריסת מערך מדידת האיכות בכל תחומי הפעילות בארגון ומקורות מידע, אחריות איסוף ואחריות דיווח.

4.4 תוצאות פרמטרי האיכות ומגמות שיפור

4.4.1 מגמות ונתונים עדכניים של כל הנתונים העיקריים המתארים את איכות המוצר, תהליך וכדי.

4.4.2 במידת האפשר, השוואה עם נתונים של המתחרים, התעשיה וכדי.

4.5 התאמה בין התוצאות בפועל לבין הדרישות

4.5.1 השיטה בה נקבעים היעדים הכמותיים בכל תחום ודרישות לקוחות, סטנדרטים, מתחרים וכדי.

4.6 שיפור בפועל בהתייעלות ופריון

4.6.1 ההישגים בפועל בנושאי התייעלות ופריון.

4.7 שילוב דיווח עלויות איכות בדוחות ונהלים של הארגון

4.7.1 תיאור קצר של מערך דיווח עלויות איכות, ניתוח הנתונים והפצת המידע.

5. תשתית לאיכות (120 נקודות)

5.1 שיטות וכלים לאיכות בפיתוח

5.1.1 מערכת נהלים להגדרת תהליך הפיתוח.

5.1.2 שילוב דרישות איכות בשלבים מוקדמים של פיתוח המוצר.

5.1.3 סקרי תיכון ואישור מעבר משלב לשלב.

5.1.4 שילוב גופי הייצור והשירות בשלבים האלה.

(*) כאשר חברה אינה עוסקת בפיתוח, לא ילקח סעיף זה בחשבון בעת הערכת החברה.

5.2 בחירת ספקים ואמצעי ייצור על בסיס שיקולי איכות

5.2.1 תהליך שיפור איכות ספקים.

5.2.2 הערכה ודירוג הספקים ושימוש בהם לבחירת הספקים על בסיס שיקולי איכות.

5.2.3 בחינת ביצועי האיכות של אמצעי ייצור והתחשבות בממצאים.

5.3 תכנון ומימוש שיטת בקרת איכות

5.3.1 מדדים ויעדי איכות בכל תחומי הפעילות בארגון.

5.3.2 שיטות בקרת התהליכים.

5.3.3 שימוש שגרתי בכלים סטטיסטיים לבקרה ושיפור תהליכים.

5.4 קיום תהליך סיקור (Audit) פנימי, שוטף וממוסד

5.4.1 תהליך סיקור - הגדרה, ארגון, שיטת פעולה, דיווח והפקת לקחים.

5.5 קיום נהלים מבוקרים, מפרטים ותקנים לבקרת איכות ושיפורה

5.5.1 נהלי איכות (Quality Manual).

5.5.2 נהלים מעודכנים ומבוקרים הנגזרים מנוהלי האיכות.

5.5.3 הפצה, בקרה ושימוש בנהלים אלה.

5.6 קיום תשתית ואמצעים מוגדרים לבקרת איכות ושיפורה

5.6.1 ארגון אבטחת איכות ושילובו במערכת הכללית של הארגון.

5.6.2 הימצאות הכלים והאמצעים הנדרשים לפעילות צוות אבטחת איכות.

5.6.3 מעקב ודיווח באיכות בכל תחומי פעילות הארגון.

5.7 מערך כיול ותחזוקה של האמצעים התפעוליים

5.7.1 מערך כיול ותחזוקה והנוהלים המגדירים את הפעילות.

5.7.2 רישום הציוד והתחזוקה, או כיול הנדרשים ולוח זמנים מחייב.

5.7.3 מערכת בקרה, דיווח וניתוח ממצאים.

6. שיפור ומיצוי המשאב האנושי (140 נקודות)

6.1 תכנון וביצוע הדרכה והכשרה בכל הדרגים

6.1.1 קביעת צורכי ההדרכה של עובדי הארגון.

6.1.2 מסגות ההדרכה בארגון (שיטה, תוכניות, אמצעים, זמן וכד׳.

6.1.3 קביעת צורכי ההדרכה ותיאור תכניות ההדרכה לאיכות של עובדי הארגון בכל הרמות.

6.1.4 קיום מעקב אחר יעילות ההדרכה, תוצאותיה, היקפה ותהליך שיפוך מתמשך.

6.2 עידוד וניצול תרומות עובדים לשיפור

6.2.1 קיום האווירה המעודדת והמסירות ליזמות ו/או שיפור על ידי העובדים.

6.2.2 תהליך הטיפול (שיטה, זמן וכד׳) בהצעות שיפור של העובדים.

6.2.3 דרכים להגדלת היקף ואיכות הצעות השיפור.

6.2.4 מדיניות ההכרה והתיגמול לעובדים שתרמו לשיפור.

6.3 עבודת צוות ומדידת אפקטיביות של מעורבות עובדים

6.3.1 האוירה והמסגרות המאפשרים מעורבות של כל העובדים בשיפוך הארגון ופעולותיו.

6.3.2 מדדים למעקב אחר היקף עבודת צוות ואת התהליך הבודק את ההשפעה על הארגון ועובדיו.

6.3.3 עידוד הארגון לעבודת צוות על ידי תִגמול אלה שתרמו לעבודת צוות.

6.4 ניהול המשאב האנושי

6.4.1 פיתוח תוכניות משאבי אנוש מתוך יעדי הארגון.

6.4.2 ·קיום אסטרטגיה ויעדים כמותיים בנושאי גיוס, הדרכה וכד׳.

6.4.3 קיום תהליך של תכנון וקידום קריירה.

6.4.4 ניתוח המידע על המשאב האנושי לשיפור והתייעלות בכל תחומי הפעילות בארגון.

6.5 הכרה בתרומה לאיכות, תגמול לאיכות

6.5.1 השתלבות מסגרות ההכרה והתגמול במדיניות האיכות.

6.5.2 הערכת התרומה לשיפור האיכות ועומת תרומה לשיפור לוח זמנים, עלויות וכד׳ ושילובה בתהליך הערכת העובדים.

6.5.3 מערכת תִגמול על שיפור באיכות, לעובדים ולצוותים ומעקב ושיפור המערכת.

6.6 עידוד השתתפות בהשתלמויות וכנסים מקצועיים

6.6.1 מדיניות עידוד השתלמות מקצועית.

6.6.2 מסגרת להכשרה מקצועית והעשרה.

6.6.3 מעקב אחר השתתפות עובדים בפעולות השתלמות והכשרה והרחבתה.

6.7 איכות חיי העבודה

6.7.1 קיום מערכת למדידת מורל העובדים.

6.7.2 מערכת קבועה לטיפול בבעיות אישיות של העובדים.

6.7.3 מערכת קבועה לשיפור רווחת הפרט.

7. איכות במידע ומערכות דיווח (80 נקודות)

7.1 תקנים ונהלים פנימיים לתיעוד.

7.2 רמת המחשוב בארגון.

7.3 עדכון, בקרה והפצה של נהלים בארגון.

7.4 ארכיון.

אינדקס תרשימים

אינדקס עברי

א

ב

ג

ד

ה

ז

ח

ט

י

כ

ל

מ

נ

ס

ע

פ

צ

ק

ר

ש

ת

אינדקס לועזי

A

B

C

D

E

F

G

K

I

J

L

M

N

O

P

Q

R

S

T

V

W

הוצאת **הוד-עמי** לספרי מחשבים

הראובני 6, ת.ד. 6108, הרצליה 46160

טלפון : 09-9564716 פקס : 09-9571582 *hod_ami@netvision.net.il*

קטלוג 3.97

לקבלת **קטלוג ממוחשב** עם כל הספרים בתמונה וצבע:

אינטרנט, חנות וירטואלית: *http://www.exlmarket.co.il/*

אינטרנט, דואר אלקטרוני: *hod_ami@netvision.net.il*

פקסס: 09-9571582 או טלפון: 09-9564716

Microsoft Press



QUE







SAMS PUBLISHING





בספרי הוד-עמי תמצא **גלויית החזר על חשבוננו**. מלא את פרטיך ושלח בדואר (אין צורך בבול). אנו נעדכן אותך בספרים חדשים, חומר עדכני, תקליטורים חדשים, מבצעים ונשלח לך את **עלון הוד-עמי למשתמשי חלונות** המכיל טיפים לעבודה בחלונות, אינטרנט ו-Word.

41 ספרים שנכתבו במקור בעברית
6 ספרים על אינטרנט
6 ספרים על מערכת הפעלה Windows 95
8 ספרים על C++
3 ספרים על Word 6
3 ספרים על Excel 5
3 ספרים על Excel 7
2 ספרים על Word 7

הוצאת הוד-עמי לספרי מחשבים הראובני 6, ת.ד. 6108 הרצליה 46160
טלפון: 09-9564716 פקס: 09-9571582 *hod_ami@netvision.net.il*

רשימת הספרים 3.97 צלצל לקבלת קטלוג חינם!!!

		עמ'		מחירי
Windows 95				
הסדרה הידידותית Windows 95	חדש	496		99
הסדרה הידידותית Word 7 for Windows 95 + CD לומדע	חדש	496	CD	119
הסדרה הידידותית Word 7 for Windows 95	חדש	496		99
הסדרה הידידותית Excel 7 for Windows 95	חדש	464		99
הסדרה הידידותית PowerPoint 7 for Windows 95	חדש	88		39
הסדרה הידידותית Excel 7 VBA for Windows 95	חדש	464		99
הסדרה הידידותית Access 7 for Windows 95	חדש	384		89
הסדרה הידידותית Internet Explorer 2 for Windows 95	חדש	224	CD	79
הסדרה הידידותית המחשב האישי שלי עם Windows 95	חדש	464	CD	99
הסדרה הידידותית HTML for Windows 95	חדש	432		85
הסדרה הידידותית Netscape 3 for Windows 95	חדש	270	CD	89
צעדים ראשונים ב- Windows 95		280	☑	79
עוברים ל- Windows 95		112		45
Windows 95 בקלות		160		59
Word 7 במשרד הממוחשב	חדש	248	☑	65
Excel 7 מהצעד הראשון + תוכנת X-Tools v2	חדש	240	☑	79
Excel 7 למי שכבר יודע קצת	חדש	304		79
Windows 95 טיפים לעבודה ברשת	חדש	256		69
Windows 95 Registry בשליטה מלאה	חדש	480	CD	139
CorelDRAW! 6 למשתמש המקצועי	חדש	768	CD	199
Qtext for Windows צעד-אחר-צעד	חדש	328	☑	79
תוכנות OFFICE לחלונות 3.11 (מתאים גם לעבודה ב-Win 95)				
Word 6 במשרד הממוחשב		240	☑	59
Word 6 בבית-הספר		144	☑	39
Word 6 צעד-אחר-צעד		560		85
Excel 5 במשרד הממוחשב + תוכנת X-Tools v2	חדש	256	☑	69
Excel 5 לא למתחילים		304		89
Access 2 פיתוח יישומים ללא קוד		432	☑	99
המחשב האישי לשירותך (חלונות, Windows95, אינטרנט, Office)	הד' 2	360	☑	79
Qtext for Windows צעד-אחר-צעד	חדש	328	☑	79
ספרי *Microsoft Press* בעברית				
Excel ספר הפונקציות		384		95
Help Desk ספר התמיכה	חדש	288		99

	עמ'	💾	מחירי
אינטרנט INTERNET			
הסדרה הידידותית Internet Explorer 2 for Windows 95 חדש	224	CD	79
הסדרה הידידותית – *Netscape 3 for Windows 95* חדש	270	CD	89
הסדרה הידידותית – *HTML for Windows 95* חדש	432		85
אינטרנט, מחפשים ב-Web	180		65
אינטרנט, גולשים עם Chameleon 4.6	256		79
המודם באוטוסטרדת המידע	320	☑	65
מערכות הפעלה (Win95 בתחילת עמוד זה)			
חלונות לקבוצות עבודה - טיפים למתחילים	320		69
חלונות לקבוצות עבודה - 1001 טיפים	968		149
מערכת הפעלה MS-DOS 6.2 - מדריך שימושי	256		45
UNIX למתחילים ויותר	480		89
PC - חומרה ותוכנה			
השבחת המחשב האישי - עשה זאת בעצמך	288		79
המחשב האישי למשתמש המקצועי (מהדורה 5)	496	☑	99
המדריך השלם לטכנאי PC ורשתות תקשורת חדש	432	☑	119
שפות תכנות			
תכנות Java עם Visual J++ חדש			
Java עכשיו חדש	336		69
משבר המחשוב של שנת 2000 חדש	424	☑	115
מבוא למדעי המחשב עם C++ חדש	504	☑	99
גרפיקה בחלונות עם Borland C++ חדש	632	CD	139
C/C++ - פונקציות ספרייה - ערכת כלים למתכנת	336	☑	79
C++ בקלות חדש הד' 2	436	☑	79
המדריך לשפת C++ ותכנות מונחה עצמים (OOP)	416	☑	89
C++ - יישומים מתקדמים	576	☑	99
ללמוד C חדש	320	☑	79
תכנות בחלונות עם C/C++ חדש	480	☑	119
המדריך השלם לשפת C (טורבו)	408	☑	79
Visual Basic Professional 3 תכנות מקצועי	360		89
בניית יישומים בעזרת Visual Basic 4 בסביבת Windows 95 חדש	388	CD	99
סדנת לימוד Visual Basic 4 חדש	632		139
שפת אסמבלי למחשב האישי	288		65
פסקל מהצעד הראשון	256		49
טורבו-פסקל - תוכניות, בעיות ופתרונות	432		59
טורבו-פסקל - תכנות מבני למתחילים ומתקדמים	400		63
מלאכת מחשב - 1 - QuickBasic	120		25
מלאכת מחשב - 2 - QuickBasic - גרפיקה וצליל	128		29
מלאכת מחשב - 3 - QuickBasic - תכנות מתקדם	208		43

		עמ'		מחיר*
מחוללי יישומים ובסיסי נתונים				
הסדרה הידידותית Access 7 for Windows 95	חדש	384		89
ארגון נתונים וקבצים		396		73
Access 2 פיתוח יישומים ללא קוד		432	☑	99
בסיסי נתונים טבלאיים ושפת SQL		480		79
כלי פיתוח תוכנה ותרגול MS-ACCESS		128		59
תקשורת				
Windows 95 טיפים לעבודה ברשת	חדש	256		69
שרת NT, Windows NT Server 3.51 - כרך א' + עדכון ל-4	חדש	240		89
שרת NT, Windows NT Server 3.51 - כרך ב' + עדכון ל-4	חדש	488		169
הכל על תקשורת ורשתות במחשב האישי	חדש	760		139
יסודות התקשוב		352		69
רשת נובל - מדריך הפעלה ושירות Novell NetWare 3.12		304		89
רשת נובל - NetWare 4.1 כרך א'		336		119
רשת נובל - NetWare 4.1 כרך ב'	חדש	450		139
רשת נובל - NetWare 4.1 כרך ג'	חדש	440		119
מעבדי תמלילים				
Qtext for Windows צעד-אחר-צעד	חדש	328	☑	79
הסדרה הידידותית Word 7 for Windows 95	חדש	496		99
Word 7 במשרד הממוחשב	חדש	248	☑	65
Word 6 במשרד הממוחשב		240	☑	59
Word 6 בבית-הספר		144	☑	39
Word 6 צעד-אחר-צעד		560		85
גיליונות אלקטרוניים				
הסדרה הידידותית Excel 7 for Windows 95	חדש	464		99
Excel 7 מהצעד הראשון + תוכנת X-Tools v2	חדש	240	☑	79
Excel 7 למי שכבר יודע קצת	חדש	304		79
Excel ספר הפונקציות		384		95
Excel 5 במשרד הממוחשב + תוכנת X-Tools v2	חדש	256	☑	69
Excel 5 לא למתחילים		304	☑	89
הגיליון האלקטרוני - משימות תרגול		112		35
ניהול				
Help Desk ספר התמיכה	חדש	288		99
משבר המחשוב של שנת 2000	חדש	424	☑	115
מערכות שרת/לקוח - טכנולוגיות ויישומים Client/Server		464		99
ניהול איכות תוכנה	חדש			

• המחירים בש"ח כולל מע"מ ומשלוח חינם.

הסדרה הידידותית

הוצאת הוד-עמי

חלונות 95



הדרך המהירה, הפשוטה והקלה לעשות דברים ב-Windows 95: עבודה עם סמלים, קבצים ותיקיות. למתחיל ולמתקדם.

6/96 496 עמ׳



הדרך המהירה, הפשוטה והקלה לעשות דברים ב-Word 7: עריכת טקסט, טבלאות, עיצוב, הדפסה, שילוב גרפיקה ותמונות ועוד.

6/96 496 עמ׳



הדרך המהירה, הפשוטה והקלה לעשות דברים ב-Excel 7: גיליונות מרהיבים, עיבוד נתונים, חישובים מורכבים, גרפיקה עסקית ועוד.

8/96 464 עמ׳

הסדרה הידידותית

הוצאת הוד-עמי



הדרך המהירה, הפשוטה והקלה לעשות דברים ב-Access 7: יצירת מסדי נתונים, עיבוד נתונים רבים ועוד.

9/96 384 עמ׳



הדרך המהירה, הפשוטה והקלה להכיר ולהתקין את רכיבי המערכת, ולהפעיל: Windows 95, מולטימדיה, אינטרנט ועוד. למנהלים.

8/96 448 עמ׳



הדרך המהירה והפשוטה לעשות דברים באינטרנט עם Netscape 3: גלישה ברשת, מולטימדיה, קניות, דואר אלקטרוני, הורדת קבצים ועוד.

צור לוין 10/96 270 עמ׳

הסדרה הידידותית

הסדרה הידידותית

Excel VBA

for Windows 95

תשובות לבעיות יומיומיות

דוגמאות צעד-אחר-צעד

טיפים וקיצורי דרך



הדרך המהירה, הפשוטה והקלה לעשות דברים ב-Excel VBA: מאקרוס, בניית יישומים ועוד. כיצד להפיק מ-Excel הרבה יותר.

12/96

הסדרה הידידותית

PowerPoint

for Windows 95

תשובות לבעיות יומיומיות

דוגמאות צעד-אחר-צעד

טיפים וקיצורי דרך



הוד-עמי



הדרך המהירה, הפשוטה והקלה לעשות דברים ב-PowerPoint 7: בניית מצגות מדהימות, שילוב גרפיקה ותמונות, אנימציה והכל בעברית.

12/96

הוצאת הוד-עמי

CorelDRAW! 6





לימוד מתקדם של התוכנה והדגשה על Windows 95. הדרכה לעבודה במולטימדיה ואינטרנט. בתקליטור: כלים, תמונות ועוד...

10/96 768 עמ׳

ניהול

Microsoft Press



Help Desk

ספר התמיכה

ספר הדרכה של Microsoft Press ללימוד תכנון, הקמה ותפעול של Help Desk.

9/96 384 עמ׳

משבר המחשוב של שנת 2000





מחשבים שומרים את השנה בשתי ספרות. 1996 זה 96 ושנת 2001 זה 01 וכאן הבעיה. איך יוצאים מזה? הכל בספר ובדיסקט.

9/96 400 עמ׳

הוצאת הוד-עמי

QTEXT

QTEXT

FOR WINDOWS

צעד-אחר-צעד

ספר לימוד עצמי למתחילים ולמתקדמים

גרסאות 6.01 ו-7.0 לחלונות 3.11 וחלונות 95



ספר ללימוד עצמי המיועד למשתמשים בכל הרמות - מתחילים ומתקדמים כאחד.

11/96 336 עמ׳

חלונות 95

הוצאת **הוד-עמי**

עוברים ל- Windows 95

צעדים ראשונים למשתמשי Windows 3.11



הספר למהגרים ולאלה שיודעים חלונות 3.11 ורוצים לעבור **מהר** לחלונות 95. קל, מהיר ופשוט.

4/96 112 עמ'



ספר למתחילים במערכת ההפעלה Windows 95. מסכים רבים, הנחיות, טיפים והערות, כדי להצליח בהפעלת התוכנה.

9/96 160 עמ'



לימוד Windows 95 על ידי 100 משימות מעשיות ומשעשעות. בצעדים פשוטים וקלים תלמדו כיצד לעבוד עם מערכת ההפעלה.

דרור מימון 7/96 280 עמ'

חלונות 95

הוצאת **הוד-עמי**

המחשב האישי לשירותך

כולל Windows 95 בעברית

ופרק מורחב על אינטרנט

הספר שכל מתחיל צריך



מיועד **לנרתעים מלגשת למחשב** בפעם הראשונה. הדרכה **צעד-אחר-צעד**: מבנה המחשב, חלונות, תוכנות פופולריות למשרד ולבית.

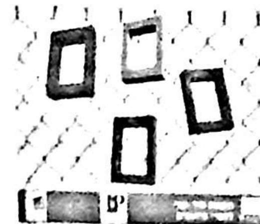
5/96 360 עמ'

Windows 95

טיפים:

לעבודה:

ברשת:



הספר מכסה את Microsoft Exchange, Mail, משלוח וקבלת פקס ו-Schedule+ - כל הכלים לעבודה ברשת.

9/96 256 עמ'



לאלה הרוצים לדעת הרבה יותר כיצד פועלת Windows 95. הבנת המבנה, היכולת והאפשרויות הגלומות ב-Registry.

10/96 480 עמ'

תוכנת WORD 6

הוצאת **הוד-עמי**



גישה מובנית
צעד-אחר-צעד, אשר
תנחה אותך משלבי
הלימוד הראשונים ועד
לשליטה מלאה בתוכנה.
הספר ל-Word 6.

1/95 560 עמ'



לעובדי המשרד הרוצים
לדעת **תכל'ס**, איך
עובדים ב-Word 6.
מתאים ללימוד עצמי,
בבית, במשרד ובכיתת
לימוד.

יהודית סלע 5/96 240 עמ'



כל מה שנדרש לעבודה
שוטפת בבית הספר
ובבית. מיועד לתלמידים
בחטיבות הביניים
ובחטיבה הגבוהה.

ויליאם פרגיון 7/96

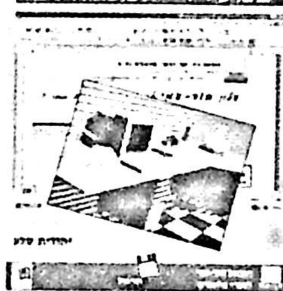
תוכנת WORD 7

הוצאת **הוד-עמי**



הדרך המהירה, הפשוטה
והקלה לעשות דברים
ב-**Word 7**: עריכת
טקסט, טבלאות, עיצוב,
הדפסה, שילוב גרפיקה
ותמונות ועוד.

6/96 496 עמ'



לעובדי המשרד הרוצים
לדעת **תכל'ס**, איך
עובדים ב-Word 7.
מתאים ללימוד עצמי,
בבית, במשרד ובכיתת
לימוד.

יהודית סלע 9/96 240 עמ'

ACCESS 2



גישה מעשית ומובנית
המנחה אותך לעיצוב
בסיסי נתונים טבלאיים.
פיתוח יישומים
מורכבים בעזרת
אובייקטים.

10/95 432 עמ'

תוכנת Excel 5

הוצאת הוד-עמי



למשתמש העובר מתוכנת Excel 4 או תוכנות אחרות. להפקת המירב מהתוכנה והגעה לרמות ביצוע שלא התנסית בהן בעבר.

10/95 304 עמ׳



ספר הדרכה ללימוד מעמיק של כל הפונקציות ב-Excel מבית *Microsoft Press*. הסברים ודוגמאות לשילוב בגיליון.

6/96 384 עמ׳



לעובדי המשרד הרוצים לדעת **תכל׳ס**, איך עובדים ב-Excel 5. מתאים ללימוד עצמי, בבית, במשרד ובכיתת לימוד.

יהודית סלע 240 עמ׳

תוכנת Excel 7

הוצאת הוד-עמי



הדרך המהירה, הפשוטה והקלה לעשות דברים ב-Excel 7 : גיליונות מרהיבים, עיבוד נתונים, חישובים מורכבים, גרפיקה עסקית ועוד.

8/96 496 עמ׳



ספר למתחילים ולעושים צעדים ראשונים באקסל בשיטת ״ראה ועשה״.

ישראל מליחי 12/96



למשתמש העובר מתוכנת Excel 4 או תוכנות אחרות. להפקת המירב מהתוכנה והגעה לרמות ביצוע שלא התנסית בהן בעבר.

9/96 304 עמ׳

אינטרנט
חלונות 3.11 ו- חלונות 95

הוצאת הוד-עמי



הוראות ברורות ומפורטות, דוגמאות רבות, מאות קטעי קוד. תכנות מכוון אובייקטים ועבודה במולטימדיה.

10/96 336 עמ׳



בשפה פשוטה וברורה תלמד בקלות ובמהירות כיצד לגלוש באינטרנט, לשלוח ולקבל דואר אלקטרוני, מולטימידה, ועוד.

צור לוין 256 עמ׳



הספר המקיף לכלי חיפוש באינטרנט בכלל וב-Web בפרט. כיצד עובדים ה-Spiders, ה-Robots וחבריהם...

לוי, עמיהוד 2/96 180 עמ׳

אינטרנט חלונות 95

הוצאת הוד-עמי



הדרך המהירה והפשוטה לעשות דברים באינטרנט עם Netscape 3 : גלישה ברשת, מולטימדיה, קניות, דואר אלקטרוני, הורדת קבצים ועוד.

צור לוין 10/96 270 עמ׳



הדרך המהירה והפשוטה לעשות דברים באינטרנט עם Internet Explorer : גלישה, מולטימדיה, קניות, דואר אלקטרוני, הורדת קבצים ועוד.

7/96 224 עמ׳



הדרך המהירה והפשוטה ליצור דפים באינטרנט מרהיבים, מדהימים ושובי עין. מולטימדיה באינטרנט!

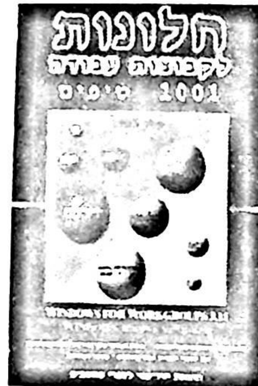
10/96 432 עמ׳

חלונות 3.11 הוצאת הוד-עמי



מאות צעדים המכילים את המרכיבים השימושיים ואת כל הפעולות הבסיסיות.

3/95 320 עמ׳



התנ״ך לחלונות. 1001 צעדים וטיפים המכילים את כל ה״יסודות״ ואת כל הפעולות שעליכם להכיר. כולל קובצי INI ועבודה ברשת.

11/94 968 עמ׳

PC למקצוענים הוצאת הוד-עמי



הדרכה צעד-אחר-צעד בלווי מאות תמונות ואיורים להתקנת חומרה ותוכנה במחשב שברשותך.

1/96 288 עמ׳



ספר יסודי ומעמיק ב״קרביים״ של המחשב האישי.
מהדורה 5 ומורחבת 1996.

מ. קליג ע. שרון 480 עמ׳



התקנות, פתרון בעיות, חומרה ותוכנה, התמודדות עם תקלות.

דורון סיון 11/96 432 עמ

שפת התכנות C++ הוצאת הוד-עמי



למהגרים משפת C ולעושים צעדים ראשונים ב- C++. דוגמאות רבות, איורים והסברים. מהדורה 2.

11/96 430 עמי



תיאור, אבות טיפוס (Prototype), ערך מוחזר של 800 פונקציות מספריית הפיתוח של Borland C/C++.

אבי בוץ 1/96 336 עמי



למידה ראשונית של עקרונות מדעי המחשב בסביבת C++. מיועד לסטודנטים ותלמידי תיכון. שפע דוגמאות, תרגילים ופתרונות.

ד"ר ע. ערמון 11/96 504

שפת התכנות C++ הוצאת הוד-עמי



למי שכבר יודע והתנסה בפיתוח יישומים ב- C++ ורוצה להתקדם לעולם האובייקטים ולנצל את אפשרויותיו המגוונות.

שמעון כהן 6/95 576 עמי



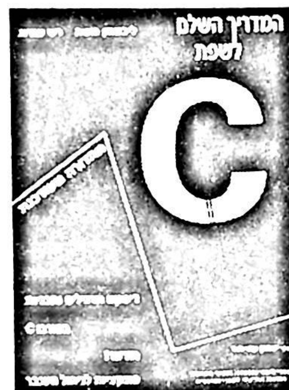
טכניקות חדשות בתכנות ב- 32 סיביות, עיצוב ממשק משתמש עדכני וריבוב משימות. מתבסס על ספריית API. בספר עשרות דוגמאות תכנות.

1/97 480 עמי



חסוך זמן תכנות ואל "תמציא את הגלגל". למד איך לנצל ביעילות קוד מעוצב כראוי הפועל ללא תקלות. כלים ועזרים לעבודה גרפית.

שפת התכנות C/C++ הוצאת הוד-עמי



ספר לימוד שפת C מהמסד ועד הטפחות המדגיש טכניקות תכנות מתקדמות.

רש, ליכטמן 408 עמ׳



לימוד הנושאים הבסיסיים בשפה בצורה עניינית. הסברים קצרים, דוגמאות ותרגילי תכנות.

יואב נתיב



הוראות ברורות ומפורטות, דוגמאות רבות, מאות קטעי קוד. תכנות מכוון אובייקטים ועבודה במולטימדיה.

10/96 336 עמ׳

שפת התכנות VB4 הוצאת הוד-עמי



פיתוח ממשקי משתמש ויישומים מתקדמים תחת Windows. גישה מובנית ללימוד צעד-אחר-צעד. גירסה 3.

5/95 360 עמ׳



מקור רב עוצמה ללימוד עקרונות ותהליכים לפיתוח יישומים: עיצוב ממשקים, OLE, הקמת קובצי Help, התקנה יישומים. גירסה 4.

12/96 370 עמ׳



מדריך ייחודי ללימוד שפת תכנות מתקדמת. מספר רב של דוגמאות העושות את ה״קשה״ לפשוט וברור. גירסה 4.

12/96

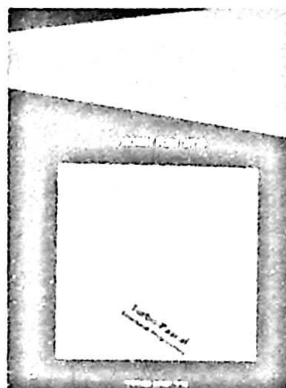
שפת התכנות פסקל

הוצאת **הוד-עמי**



לתלמידים המתחילים בלימוד השפה. מותאם ל-3 יחידות לימוד.
על פי תוכנית הלימודים.

ויליאם פרג׳ון 256 עמ׳



ספר לימוד מקיף לתלמידים. מותאם ל-5 יחידות לימוד.
על פי תוכנית הלימודים.

אבא אנגלברג 400 עמ׳



לימוד בדרך של ניסוי, תרגול ותשובות. המשך ישיר לספר **פסקל - תכנות מבני**. 200 שאלות ותרגילים.

רון בורדו 432 עמ׳

שפות תכנות

הוצאת **הוד-עמי**



3 ספרים בסדרה יחידה במינה, ללימוד עצמי של Quick Basic בדרך של התנסות ופתרון משימות תכנות.

דרור מימון



ספר מפורט על פקודות שפת אסמבלי למעבדי אינטל 8088/8086 : דוגמאות, תרגילים עם פתרונות מפורטים.

אלי כהן 11/93 288 עמ׳

UNIX



מדריך מקיף למשתמש המתחיל ולמי שרוצה להרחיב את ידיעותיו. הדרכה מפורטת צעד-אחר-צעד.

5/95 480 עמ׳

רשתות תקשורת

הוצאת הוד-עמי



ספר חובה בידיו של כל מנהל רשת ומשתמש! מותאם עד גירסה 3.12.

10/94 304 עמ׳



3 כרכים המרכיבים יחד את הספר השלם ביותר לרשת נובל. מיועד למנהלי רשת, מקימי רשתות ולמשתמש המתוחכם.

8/96 1226 עמ׳



2 כרכים המהווים יחד את הספר המקיף ביותר להתקנה, הפעלה ואחזקה של שרת NT, NT Server 3.51. התאמה ל- NT 4.0 .

8/96 680 עמ׳

תקשורת

הוצאת הוד-עמי



מותאם לתוכנית הלימודים ומיועד לסטודנטים, תלמידים ומנתחי מערכות.

גדעון קוך 11/95 352 עמ׳



ספר מקיף על עולם התקשורת: חומרה, תוכנה וטכנולוגיות עם דגש על PC ושילובו ברשתות מסוגים שונים.

4/96 760 עמ׳



מדריך ייחודי ומקיף למערכות שרת/לקוח: חומרה, תוכנה, טכנולוגיות ויישומים.

6/95 464 עמ׳

כשחושבים מחשבים...



קוראים **הוד-עמי.**

הוצאת הוד-עמי לספרי מחשבים, הגדולה והמובילה בישראל
מציעה לכם מגוון מקצועי ומעודכן של ספרי הדרכה למחשבים:
Windows 95, אינטרנט, תוכנות Microsoft, שפות תכנות ועוד.



הוד-עמי ספרים שמדברים אליך

DoubleClick

ניהול איכות תוכנה

מה זה, ולשם מה?

ניתוח מערכות והנדסת תוכנה מבטאים את העבודה המעשית הכרוכה בפיתוח מערכת תוכנה. אולם, מהי דמותה של המערכת ומה איכותה ויכולתה לעמוד במבחן המציאות?

הדרך היחידה למסור ללקוח מערכת איכות, הינה באמצעות **ניהול איכות תוכנה-** Software Quality Management. עבודה על פי כללי איכות הינה "דרך חיים" שיש לה משמעות ותוקף עבור הלקוח וגם עבור מפתח התוכנה, כדי להשיג תוצאות אפקטיביות.

הספר מציג את גישת **ניתוח המערכת** מתוך ההיבט של ניהול האיכות באמצעות תקני ISO-9001 ו- ISO-9000-3. הוא הופך את התקן הפורמלי לכלי עבודה מעשי ועושה אותו נגיש ומובן לכל מתכנן, מעצב ומתכנת.

הספר גם מציג שני נושאים מרכזיים: **מדידה**, המאפשרת שיפור בעתיד מתוך השוואה למצב קיים; **וטכניקות איטרטיביות ומבניות** בתחום הנדסת התוכנה.
תוכל גם לראות כיצד ליצור סביבת פיתוח משולבת שתתרום לתוצאות, ולא תכביד על הארגון.

מכון התקנים הישראלי הסב את תקני ISO-9001 ו- ISO-9000-3 לסביבת העבודה בישראל, וכינה אותם **ת"י 2001** ו- **ת"י 3-2000** בהתאמה.

נספח א' כולל את ת"י 3-2000 במלואו.

מה תמצא בספר?

- הנחיות מעשיות להבנת תקני ISO-9001 ו- ISO-9000-3 ויישומם.
- מבט מקיף והדגשת מהלכי המדידה של שלבי הפיתוח.
- שיקולים והבחנה בין תהליכי פיתוח איטרטיביים ומבנים והשלכותיהם על המוצר הסופי - מערכת התוכנה.
- ניתוח ההשלכות של תהליכי פיתוח חפוזים ושל פיתוח מוכוון עצמים על פי כללי האיכות.

המחבר, **בריאן המבלינג**, הינו מהנדס אלקטרוניקה שהתמחה בפיתוח תוכנה.
עוסק בניהול צוותי פיתוח מערכות תוכנה ומוסמך TickIT. מדריך ומנחה יועצי פיתוח מערכות ומהנדסי תוכנה בתחומי ניהול איכות.

מחיר לצרכן: 129 ש"ח



ISBN 965-361-124-0 מסת"ב